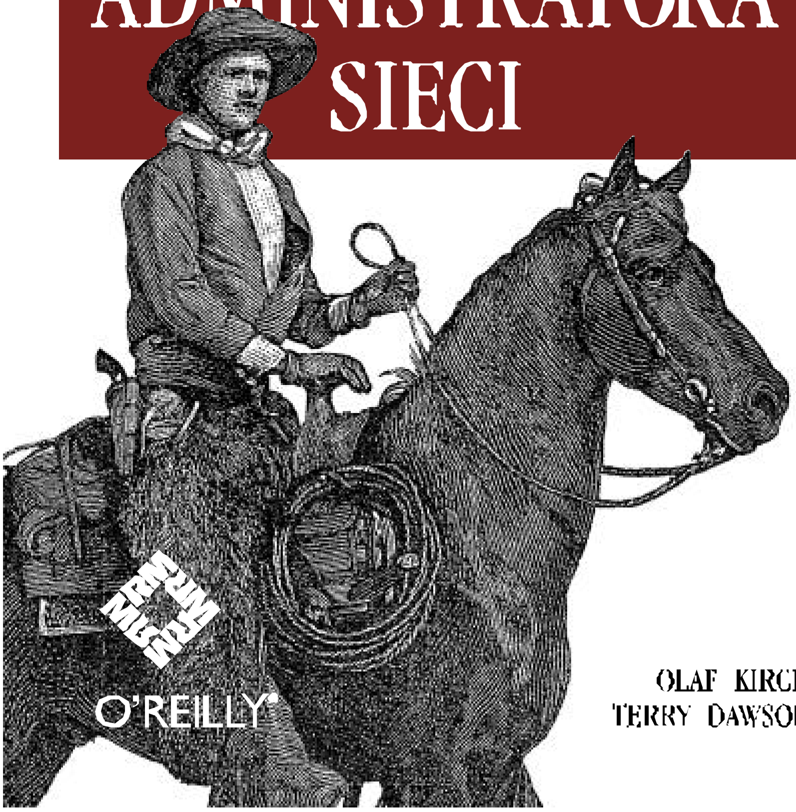
LINUX

PODRĘCZNIK ADMINISTRATORA SIECI



O'REILLY®

OLAF KIRCH
TERRY DAWSON

Linux – podręcznik administratora sieci

Olaf Kirch, Terry Dawson

Tytuł oryginału: Linux Network Administrator's Guide, Second Edition

Tłumaczenie: Krzysztof Łabanowski

Wydawnictwo RM, Warszawa 2000

Wydawnictwo RM, 00-987 Warszawa 4, skr. poczt. 144
rm@rm.com.pl
www.rm.com.pl

ISBN 83-7243-116-7

Redaktor prowadzący: Danuta Cyrul

Redakcja: Irmina Pęgierska
Korekta: Mirosława Szymańska

Opracowanie graficzne okładki według oryginału: Grażyna Jędrzejec
Redaktor techniczny: Beata Donner-Soska
Skład: Marcin Fabijański
Druk i oprawa: Oficyna Wydawnicza READ ME – Drukarnia w Łodzi

Wydanie I

10 9 8 7 6 5 4 3 2 1

Spis treści

Wstęp



Termin „Internet” zadomowił się już na dobre w wielu językach, a mnóstwo, skądinąd po wa żnych lu dzi, z ra do ścią po dr óżuje po in fo stra dzie. Dlatego mo żna po wie dzieć, że sie ci kom pu te ro we stają się już czy mś tak zwykłym jak te le wi zo ry i ku chen ki mi kro fa lo we. In ter net cie szy się nie zwy kłym za in te re so wa niem me diów, a stu den ci so cjo lo gii za czy na ją się spe cja li zo wać w gru pach dys ku syj nych Usenetu, środowiskach elektronicznej rzeczywistości wirtualnej i WWW, badając w ten sposób nową „kul tu rę in ter ne tową” .

Oczy wiś cie sieć ist nieje z da wien daw na. Łącze nie kom puterów tak, aby two rzyły sie ci lo kal ne, było po wszech ne w przy pad ku małych in sta la cji, a gdy ma szy ny były od siebie oddalone wykorzystywano łącza telekomunikacyjne. Jednakże szybki rozw ój sie ci ogó lnoświatowych dał szan sę przyłącze nia się do glo bal nej wio ski wielu zwykłym użyt kow ni kom komp ute rów oraz małym, nie do cho do wym or ga ni zacjom pry wat nym. Wy ra źnie spa dają ce ny ho stów in ter ne to wych z obsługą pocz ty i grup dys ku syj nych przez do stęp ko mu to wa ny oraz ISDN, a po ja wie nie się DSL (*Digital Subscriber Line*) oraz techno lo gii mode mów ka blo wych niewątpli wie podtrzy ma tę ten den cję.

Jeżeli mówimy o sieciach komputerowych, nie sposób nie wspomnieć o Uniksie. Oczywiście Unix nie jest jedynym systemem operacyjnym, który może pracować w sie ci, ani też na wet nie jest naj po pu lar niej szym z nich, ale w biz ne sie sie cio wym ist nie je od daw na i z całą pew no ścią bę dzie ist niał jesz cze przez ja kiś czas.

Unix jest szczególnie cie kawy dla zwykłych użytkowników dzię ki te mu, że włożono wie le wy siłku w stwo rzen ie dla PC dar mow ych uniks owy ch syste mów ope ra cyjnych, ta kich jak 386BSD, Fre eBSD czy Li nux.

Li nux jest dys try bu owaną bezpłat nie od mianą Unik sa, prze zna czoną dla kom pu terów oso bi stych. Ak tu al nie działa na róż nych ma szy nach, i na tych z pro ce sa mi fir my In tel, z pro ce so ra mi Mo to ro la 680x0 (np. Com mo do re Ami ga i Ap ple Ma cin tosh); na maszynach Sun SPARC i Ultra-SPARC; na Alphach firmy Compaq; MIPS-ach; na PowerPC, czy li na no wej ge ne ra cji kom pu te rów Ap ple Ma cin tosh, i na Strong-

ARM-ach, takich jak Netwinder firmy rebel.com czy palmtopy firmy 3Com. Linux został zaadaptowany także na pewne stosunkowo mało znane platformy, takie jak Fujitsu AP-1000 i IBM System 3/90. Aktualnie realizowane są adaptacje na inne interesujące architektury, a zadanie przeniesienia Linuksa do postaci zamkniętego kontrolera także wygląda obiecująco.

Linux rozwija się dzięki zaangażowaniu dużej grupy ochotników z Internetu. Projekt został zapoczątkowany w 1990 roku przez Linusa Torvaldsa – wówczas studenta fińskiego college'u – w ramach zajęć z systemów operacyjnych. Od tego czasu Linux urósł do rangi pełnego klonu Uniksa, na którym można uruchamiać aplikacje tak różnorodne, jak programy do symulacji i modelowania, procesory tekstu, systemy rozpoznawania mowy, przeglądarki WWW i mnóstwo innego oprogramowania, włączając w to różne doskonałe gry. Współpracuje z różnorodnym sprzętem, a zawiera pełną implementację sieci TCP/IP (łącznie z protokołami SLIP i PPP oraz firewallami), pełną implementację protokołu IPX, a także implementacje wielu funkcji oraz protokołów, których nie znajdziemy w żadnym innym systemie operacyjnym. Linux jest wydajny, szybki i darmowy, a jego popularność na świecie poza Internetem rośnie w szybkim tempie.

Sam system operacyjny Linux został objęty licencją publiczną GNU, tą samą, która jest używana przez oprogramowanie tworzone przez Free Software Foundation (Fundację Wolnego Oprogramowania). Licencja pozwala każdemu na dystrybuowanie i modyfikowanie oprogramowania (bezpłatnie lub dla zysku) dopóty, dopóki wszystkie modyfikacje i dystrybucje są również bezpłatnie udostępniane. Określenie „wolne oprogramowanie" oznacza wolność aplikacji, a nie wolność kosztów.

Po co i dla kogo jest ta książka

Niniejsza książka została napisana po to, aby w jednym miejscu zebrać informacje potrzebne administratorom sieci środowiska Linux. Zarówno początkujący, jak i zaawansowani użytkownicy powinni tu znaleźć informacje potrzebne do wykonania większości najważniejszych zadań administracyjnych, wymaganych do konfiguracji sieci w Linuksie. Temat tej książki – sieci – jest prawie nieograniczony, a więc oczywiście niemożliwością jest opisanie wszystkiego i w każdym aspekcie. Podjęliśmy próbę prezentacji większości ważnych i powszechnie spotykanych zadań. Naszym zamierzeniem było, aby ta książka służyła pomocą nawet początkującym adeptom sieci linuksowych (także tym, którzy nie mieli jeszcze do czynienia z uniksopodobnym systemem operacyjnym), aby po jej lekturze mogli poprawnie skonfigurować swoją sieć w Linuksie.

Istnieje wiele książek i innych źródeł informacji, które poruszają tematy opisane w tej książce (z małymi wyjątkami prawdziwie linuksowych funkcji, takich jak nowy interfejs firewalla, który nie jest nigdzie indziej dobrze udokumentowany). Gdybyś chciał się dowiedzieć więcej, w poniższym podrozdziale zamieszczamy bibliografię.

Zródła informacji

Jeżeli jesteś nowicjuszem w świecie Linuksa, masz wiele do przejrzenia i przeczytania. Pomocne, aczkolwiek niekonieczne, jest posiadanie dostępu do Internetu.

Podręczniki ze społu Linux Documentation Project (Projekt Dokumentacji Linuksa – LDP*)*

Projekt Dokumentacji Linuksa to grupa ochotników, którzy opracowują książki (przewodniki), dokumenty HOWTO, strony podręcznika elektronicznego na różne tematy: od instalacji po programowanie jądra. Publikacje LDP to między innymi:

Linux Installation and Getting Started

Ta książka, napisana pod kierunkiem Matta Welsha, opisuje jak zdobyć, zainstalować i używać Linuksa. Zawiera wprowadzenie do Uniksa i informacje o administracji systemu, systemie X Window oraz sieci.

Linux System Administration Guide

Ta książka, napisana przez Larsa Wirzeniusa i Joannę Oja, jest ogólnym przewodnikiem po administracji Linuksa i porusza takie tematy, jak tworzenie i konfigurowanie użytkowników, wykonywanie kopii zapasowych systemu, konfigurowanie podstawowych pakietów i instalowanie oraz uaktualnianie oprogramowania.

Linux System Administration Made Easy

Ta książka, napisana przez Steve'a Framptona, opisuje codzienne zadania administracyjne i zagadnienia związane z utrzymaniem Linuksa w odniesieniu do jego użytkowników.

Linux Programmers Guide

Ta książka, napisana przez B. Scotta Burketta, Svena Goldta, Johna D. Harpera, Svena van der Meera i Matta Welsha, będzie interesująca dla tych, którzy chcą tworzyć aplikacje dla Linuksa.

The Linux Kernel

Ta książka, napisana przez Davida A. Ruslinga, zawiera wprowadzenie do jądra Linuksa: opisuje jego budowę oraz działanie.

The Linux Kernel Module Programming Guide

Ta książka, napisana przez Ori Pomerantza, stanowi podręcznik wyjaśniający, jak pisać moduły jądra Linuksa.

W fazie tworzenia są kolejne podręczniki. Więcej informacji na temat LDP znajdziesz na stronach WWW pod adresem *http://www.linuxdoc.org/* lub jednym z jego serwerów lustrzanych.

Dokumenty HOWTO

Dokumenty HOWTO poświęcone Linuksowi to szereg szczegółowych opracowań omawiających bardzo różne aspekty systemu, takie jak instalacja i konfiguracja oprogramowania systemu X Window lub pisanie w asemblerze pod Linuksem. Generalnie znajdują się one w podkatalogu *HOWTO* ośrodków FTP lub są dostępne na stronach WWW zawierających dokumenty Projektu Dokumentacji Linuksa. W pliku *HOWTO-INDEX* znajdziesz listę tego, co jest dostępne.

Mogą ci się przydać: *Installation HOWTO*, opisujący jak zainstalować Linuksa na twoim komputerze, *Hardware Compatibility HOWTO*, zawierający listę urządzeń, o których wiadomo, że działają w Linuksie, oraz *Distribution HOWTO*, zawierający listę sprzedawców oprogramowania oferujących Linuksa na dyskietkach lub na płytach CD-ROM.

Często zadawane pytania na temat Linuksa (Linux Frequently Asked Questions), FAQ

FAQ (*The Linux Frequently Asked Questions with Answers*) gromadzi różnorodne pytania i odpowiedzi na temat systemu. Jest to obowiązkowa lektura dla każdego nowicjusza.

Dokumentacja dostępna przez FTP

Jeżeli masz dostęp do anonimowych serwerów FTP, możesz z nich pobrać całą wspomnianą tutaj dokumentację Linuksa. Wypróbuj takie adresy jak *metalab.unc.edu:/pub/Linux/docs* i *tsx-11.mit.edu/pub/linux/docs*.

Dokumentacja dostępna przez WWW

Dostępnych jest wiele ośrodków WWW związanych z Linuksem. Macierzysta strona Projektu Dokumentacji Linuksa znajduje się pod adresem *http://www.linuxdoc.org/*.

OSWG (*Open Source Writers Guild*) jest projektem wykraczającym poza Linuksa. OSWG, podobnie jak ta książka, opowiada się za tworzeniem dokumentacji OpenSource. Witryna macierzysta OSWG znajduje się pod adresem *http://www.oswg.org:8080/oswg*.

Obie powyższe witryny zawierają wersje hipertekstowe (i inne) wielu dokumentów związanych z Linuksem.

Dokumentacja dostępna odpłatnie

Liczne wydawnictwa i sprzedawcy oprogramowania publikują prace stworzone w ramach Projektu Dokumentacji Linuksa. Dwaj przykładowi sprzedawcy to:

Specialized Systems Consultants, Inc. (SSC)
http://www.ssc.com/
P.O. Box 55549 Seattle, WA 98155-0549
1-206-782-7733
1-206-782-7191 (faks)
sales@ssc.com

oraz

Linux Systems Labs
http://www.lsl.com/
18300 Tara Drive
Clinton Township, MI 48036
1-810-987-8807
1-810-987-3562 (faks)
sales@lsl.com

Obie firmy sprzedają kompendia dokumentów HOWTO i innej dokumentacji dotyczącej Linuksa w formie drukowanej.

O'Reilly & Associates wydaje serię książek o Linuksie. Niniejsza książka powstała w ramach Projektu Dokumentacji Linuksa, ale większość została napisana niezależnie. Należą do nich:

Running Linux (wyd. pol.: *Linux*, Wydawnictwo RM, Warszawa 2000)
: Przewodnik po instalacji i użytkowaniu systemu, opisujący, jak najlepiej wykorzystać komputer osobisty, pracując w Linuksie.

Learning Debian GNU/Linux

Learning Red Hat Linux (wyd. pol.: *Red Hat Linux*, Wydawnictwo RM, Warszawa 2000)
: Książki bardziej podstawowe niż *Running Linux*. Zawierają one popularne dystrybucje na płycie CD-ROM i informują dokładnie, jak je skonfigurować i jak z nich korzystać.

Linux in Nutshell (wyd. pol.: *Linux – podręcznik użytkownika*, Wydawnictwo RM, Warszawa 1999)
: Kolejna książka z doskonałej serii „podręcznik użytkownika". Daje wyczerpujący opis poszczególnych poleceń Linuksa.

Linux Journal and Linux Magazine

„Linux Journal" i „Linux Magazine" to miesięczniki dla społeczności linuksowej, pisane i wydawane przez licznych linuksowych aktywistów. Poziom artykułów jest bardzo różny: od pytań nowicjuszy, po odpowiedzi dotyczące programowania jądra. Nawet jeżeli masz dostęp do grup dyskusyjnych Usenetu, te czasopisma są doskonałym sposobem, aby być na bieżąco ze sprawami społeczności Linuksa.

„Linux Journal" jest najstarszym czasopismem i jest wydawany przez wspomniane wcześniej SSC, Incorporated. Czasopismo to możesz także znaleźć w sieci WWW pod adresem *http://www.linuxjournal.com/*.

„Linux Magazine" jest nowszą, niezależną publikacją. Macierzysty adres WWW tego czasopisma to *http://www.linuxmagazine.com/*.

Linuksowe grupy dyskusyjne Usenetu

Oto grupy dyskusyjne Usenetu poświęcone Linuksowi:

comp.os.linux.announce
: Moderowana grupa dyskusyjna zawierająca zapowiedzi nowego oprogramowania, dystrybucji, raporty o błędach i nowinki z życia społeczności Linuksa. Wszyscy użytkownicy Linuksa powinni czytać tę grupę. Propozycje mogą być wysyłane na adres *linux-announce@news.ornl.gov*.

comp.os.linux.help
: Ogólne pytania i odpowiedzi na temat instalacji i użytkowania Linuksa.

comp.os.linux.admin
: Dyskusje związane z administrowaniem systemu Linux.

comp.os.linux.networking
  Dyskusje związane z siecią w Linuksie.

comp.os.linux.development
  Dyskusje na temat tworzenia jądra Linuksa i samego systemu.

comp.os.linux.misc
  Inne dyskusje, które nie pasują do żadnej z poprzednich kategorii.

Istnieje również kilka innych grup poświęconych Linuksowi i prowadzonych w językach innych niż angielski, a należą do nich na przykład *fr.comp.os.linux* po francusku czy *de.comp.os.linux* po niemiecku.

Pocztowe listy dyskusyjne związane z Linuksem

Istnieje szereg specjalistycznych pocztowych list dyskusyjnych na temat Linuksa. Spotkasz na nich wiele osób, które chętnie odpowiedzą na twoje pytania.

Najbardziej znane z nich to listy obsługiwane przez uniwersytet Rutgers. Możesz się do nich zapisać, wysyłając wiadomość e-mail sformatowaną w następujący sposób:

```
To: majordomo@vger.rutgers.edu
Subject: anything at all
Body:
subscribe nazwa-listy
```

Niektóre listy związane z siecią w Linuksie to:

linux-net
  Dyskusje związane z siecią w Linuksie.

linux-ppp
  Dyskusje związane z implementacją PPP w Linuksie.

linux-kernel
  Dyskusje związane z tworzeniem jądra Linuksa.

Elektroniczne wsparcie Linuksa

W wielu miejscach w sieci można uzyskać pomoc elektroniczną. Ochotnicy z całego świata oferują tam swoją specjalistyczną wiedzę i usługi tym użytkownikom, którzy mają pytania i problemy.

Sieć Open Projects IRC to sieć IRC poświęcona w całości projektom otwartym – zarówno Open Source, jak i Open Hardware. Niektóre kanały są przeznaczone do udostępniania elektronicznego wsparcia dla Linuksa. IRC to skrót od *Internet Relay Chat*. Jest to usługa sieciowa pozwalająca interaktywnie „rozmawiać" przez Internet z innymi użytkownikami. Sieci IRC obsługują wiele kanałów, na których grupy prowadzą pisane „rozmowy". Cokolwiek napiszesz na kanale, będzie to widoczne dla wszystkich pozostałych uczestników „rozmowy".

W sieci Open Projects IRC istnieje szereg aktywnych kanałów, na których spotkasz użytkowników przez 24 godziny na dobę, 7 dni w tygodniu. Są to użytkownicy, któ-

rzy chcą i potrafią pomóc w rozwiązaniu twoich problemów z Linuksem albo mogą po prostu z tobą pogadać. Z usługi tej możesz korzystać po zainstalowaniu klienta IRC, na przykład *irc-II*, podłączeniu się do serwera o zadanej nazwie, np. **irc.openprojects.org:6667,** i przyłączeniu się do kanału `#linpeople`.

Grupy użytkowników Linuksa

Bezpośrednią pomoc oferuje też wiele grup użytkowników Linuksa z całego świata. Ich uczestnicy angażują się w taką działalność, jak organizowanie dni instalacji, seminaria i dyskusje panelowe, prezentacje i inne imprezy towarzyskie. Grupy użytkowników Linuksa są doskonałym sposobem na spotkanie się z innymi linuksowcami z twojego rejonu. Istnieje szereg list grup użytkowników Linuksa. Do lepiej znanych należą:

Group of Linux Users Everywhere – http://www.ssc.com/glue/groups
LUG list project – http://www.nllgg.nl/lugww/
LUG registry – http://www.linux.org/users/

Skąd wziąć Linuksa

Nie ma jednej jedynej dystrybucji oprogramowania dla Linuksa. Takich dystrybucji jest wiele, m.in. Debian, Red Hat, Caldera, Corel, SuSE i Slackware. Każda dystrybucja zawiera wszystko, czego potrzebujesz do uruchomienia pełnego systemu Linux: jądro, podstawowe programy użytkowe, biblioteki, pliki pomocnicze i aplikacje.

Dystrybucje Linuksa można zdobyć z szeregu źródeł elektronicznych, jak Internet. Każda poważna dystrybucja posiada własny ośrodek FTP i WWW. Oto niektóre ośrodki:

Caldera
http://www.caldera.com/ftp://ftp.caldera.com/

Corel
http://www.corel.com/ftp://ftp.corel.com/

Debian
http://www.debian.org/ftp://ftp.debian.org/

RedHat
http://www.redhat.com/ftp://ftp.redhat.com/

Slackware
http://www.slackware.com/ftp://ftp.slackware.com/

SuSE
http://www.suse.com/ftp://ftp.suse.com/

Popularne archiwa FTP również zawierają różne dystrybucje Linuksa. Najbardziej znane z nich to:

metalab.unc.edu:/pub/Linux/distributions/
ftp.funet.fi:/pub/Linux/mirrors/
tsx-11.mit.edu:/pub/linux/distributions/
mirror.aarnet.edu.au:/pub/linux/distributions/

Każda dystrybucja zawiera pewne podstawowe biblioteki, narzędzia konfiguracyjne, aplikacje systemowe i pliki konfiguracyjne. Niestety, różnice pomiędzy wersjami, nazwami i lokalizacjami powodują, że bardzo trudno jest zgadnąć, co będzie w danej dystrybucji. A bez tej wiedzy nie da się stworzyć binarnych wersji aplikacji, które działałyby niezawodnie we wszystkich dystrybucjach Linuksa.

Aby rozwiązać ten problem, powołano nowy projekt o nazwie „Linux Standard Base" (standardowa podstawa Linuksa). Jego celem jest opisanie standardowej podstawy dystrybucji, do której dostosują się poszczególne dystrybucje. Jeżeli programista stworzy aplikację w oparciu o standardową podstawę, to będzie ona działała we wszelkich dystrybucjach zgodnych ze standardem.

Informacje na temat stanu projektu standardowej podstawy Linuksa możesz znaleźć na jego stronie macierzystej pod adresem *http://www.linuxbase.org/.*

Jeżeli martwisz się o zgodność, szczególnie oprogramowania komercyjnego, powinieneś upewnić się, czy w przypadku twojej dystrybucji zostały podjęte kroki prowadzące do zgodności z projektem standaryzacyjnym.

O tej książce

Gdy Olaf dołączył do Projektu Dokumentacji Linuksa w 1992 roku, napisał dwa małe rozdziały na temat UUCP i *smaila*, które zamierzał umieścić w książce *System Administrator's Guide*. Sieci TCP/IP zaczęły dopiero powstawać. W miarę ich rozwoju te dwa „małe rozdziały" zaczęły się rozrastać. Wtedy Olaf pomyślał, że byłoby dobrze mieć przewodnik po sieci. Każdy mówił: „Świetny pomysł", „zrób to!". A więc wziął się do pracy i napisał pierwszą wersję przewodnika po sieci, która została wydana we wrześniu 1993 roku.

Olaf kontynuował prace nad przewodnikiem po sieci i ostatecznie stworzył znacznie rozszerzoną jego wersję. Rozdział na temat *sendmaila* napisał Vince Skahan. W tym wydaniu rozdział ten został całkowicie zmieniony, ze względu na nowy interfejs konfiguracyjny *sendmaila*.

Wersja książki, którą czytasz, została skorygowana i uaktualniona przez Terry'ego Dawsona* na życzenie wydawnictwa O' Reilly & Associates. Terry przez 20 lat był operatorem radiaamatorskiego, z czego 15 lat przepracował w przemyśle telekomunikacyjnym. Był współautorem dokumentu NET-FAQ i napisał oraz utrzymywał różne dokumenty HOWTO związane z siecią. Terry zawsze z entuzjazmem wspierał projekt podręcznika administratora sieci i dodał w niniejszej edycji kilka rozdziałów na najnowsze tematy, które ze zrozumiałych względów nie trafiły do pierwszego wydania. Dokonał też mnóstwa zmian w celu uaktualnienia całej książki.

Rozdział omawiający *exim* napisał Philip Hazel**, który jest głównym twórcą pakietu.

---

* Z Terrym Dawsonem można się skontaktować pod adresem *terry@linux.org.au.*

** Z Philipem Hazelem można skontaktować się pod adresem *ph10@cus.cam.ac.uk.*

Książka ta ma formę sekwencji kroków, jakie należy podjąć, by skonfigurować system do pracy w sieci. Rozpoczyna się omówieniem podstawowych pojęć sieciowych, a w szczególności sieci opartych na TCP/IP. Następnie kolejno wprowadza w konfigurowanie TCP/IP na poziomie urządzenia konfigurowanie firewalli, liczenie ruchu IP (accounting) i maskowanie IP, wreszcie w konfigurowanie popularnych aplikacji, takich jak *rlogin* i tym podobne, sieciowego systemu plików (NFS – *Network File System*) oraz systemu informacji sieciowej (NIS – *Network Information System*). Dalej znajduje się rozdział o tym, jak skonfigurować maszynę jako węzeł UUCP. Większość pozostałych podrozdziałów jest poświęcona dwóm podstawowym aplikacjom, które działają na TCP/IP i UUCP: poczcie elektronicznej i grupom dyskusyjnym. Specjalny rozdział został poświęcony protokołowi IPX i systemowi plików NCP, ponieważ są one używane w środowiskach korporacyjnych, w których spotyka się Linuksa.

W części omawiającej pocztę znajduje się bardziej gruntowne wprowadzenie do transportu i rutingu poczty oraz miriady schematów adresowania, które możesz napotkać. Opisuje ona konfigurację *exima* i zarządzanie nim. Exim to agent transportowy poczty, idealny tam, gdzie nie są wymagane UUCP ani tym bardziej *sendmail*, który jest dla realizujących ruting bardziej skomplikowany, niż te obsługiwane przez UUCP.

Część poświęcona grupom dyskusyjnym daje pojęcie o tym, jak działa Usenet. Omawia INN i C News – dwa powszechnie używane pakiety oprogramowania transportowego grup dyskusyjnych oraz zastosowanie NNTP do zapewnienia dostępu do czytania grup w sieci lokalnej. Książkę zamyka rozdział na temat stosowania najpopularniejszych programów do czytania grup dyskusyjnych w Linuksie.

Oczywiście książka ta na pewno nie jest w stanie wyczerpująco odpowiedzieć na wszystkie potencjalne pytania. Tak więc, jeżeli będziesz postępował zgodnie z instrukcjami w niej zawartymi, a coś wciąż nie będzie działało, bądź cierpliwy. Niektóre z twoich problemów mogą wynikać z naszych błędów (zobacz podrozdział *Zgłaszanie uwag* w dalszej części *Wstępu*), ale mogą także być spowodowane zmianami w oprogramowaniu sieciowym. Dlatego powinieneś sprawdzić najpierw informacje zawarte w zasobach. Istnieje duże prawdopodobieństwo, że nie tylko ty masz takie problemy, a więc poprawka lub przynajmniej proponowane rozwiązanie jest już być może znane. Jeżeli masz okazję, powinieneś także spróbować zdobyć najnowszą wersję jądra i sieci z jednego z linuksowych ośrodków FTP lub z pobliskiego BBS-u. Wiele problemów wynika z nierównomiernego rozwoju różnego oprogramowania, które nie współpracuje poprawnie ze sobą. W końcu Linux to „praca w toku".

Oficjalna wersja drukowana

Na jesieni 1993 roku Andy Oram, który prawie od początku był związany z listą dyskusyjną LDP, zaproponował Olafowi opublikowanie tej książki w wydawnictwie O' Reilly & Associates. Był nią zachwycony, ale nigdy nie przypuszczał, że odniesie ona taki sukces. Postanowiono, że O'Reilly stworzy rozszerzoną oficjalną wersję drukowaną przewodnika po sieci, natomiast Olaf zatrzyma prawa autorskie i źródła

książki będą mogły być rozpowszechniane za darmo. Oznacza to, że masz wolny wybór: możesz wziąć różne darmowe wersje dokumentu z najbliższego ośrodka lustrzanego Projektu Dokumentacji Linuksa i wydrukować je sobie albo zakupić oficjalną wersję drukowaną wydaną przez O'Reilly'ego.

Nasuwa się pytanie: dla czego masz płacić za coś, co możesz mieć za darmo? Czy Tim O'Reilly postradał zmysły i wydaje coś, co każdy może sobie sam wydrukować, a nawet sam sprzedawać?*. Czy istnieją jakieś różnice pomiędzy tymi wersjami?

Od powiedzi brzmią „to zależy", „nie, zdecydowanie nie" i „tak i nie". O'Reilly & Associates podejmuje ryzyko, wydając przewodnik po sieci w formie tradycyjnej, ale jakoś się im to opłaca (poprosili autorów, by przygotowali następne wydanie). Wierzymy, że to przedsięwzięcie jest doskonałym przykładem tego, jak świat darmowego oprogramowania i firmy komercyjne mogą ze sobą współpracować, by stworzyć coś, z czego obie strony czerpią korzyści. Z naszego punktu widzenia wydawnictwo O'Reilly przysłużyło się społeczności Linuksa (nie tylko tą książką, która jest dostępna w twojej księgarni). Dzięki niemu Linux zaczął być rozpoznawany jako coś poważnego: jako rentowna i użyteczna alternatywa dla innych, komercyjnych systemów operacyjnych. Jeżeli jakaś księgarnia techniczna w USA nie ma u siebie przynajmniej jednej półki z książkami wydawnictwa O'Reilly, to jest to kiepska księgarnia.

Dla czego to wydają? Uznają to za swoją specjalność. Oto, czego oczekują, podpisując z autorami kontrakt na napisanie książki o Linuksie: tempo, poziom szczegółowości i styl mają dokładnie odpowiadać innym wydanym przez nich książkom.

Celem licencji LDP jest zapewnienie wszystkim dostępu do książki. Niektórzy mogą wydrukować sobie tę książkę sami i nikt nie będzie cię winił, jeżeli z niej skorzystasz. Jednak, jeżeli nie miałeś okazji zobaczyć wersji wydawnictwa O'Reilly, przejdź się do księgarni albo obejrzyj książkę u kolegi. Wydaje nam się, że spodoba ci się to, co zobaczysz, i będziesz chciał książkę kupić.

Jakie są więc różnice pomiędzy wersją drukowaną a wersją elektroniczną? Andy Oram włożył wiele pracy w to, aby przetworzyć nasze chaotyczne myśli w porządny wykład wart wydrukowania. (Dokonał także korekty kilku innych książek stworzonych w ramach Projektu Dokumentacji Linuksa, służąc społeczności Linuksa całą swoją fachową wiedzą).

Na redakcji Andy'ego książka znacznie zyskała w stosunku do wersji oryginalnej. Nie można było marnować okazji skorzystania z usług i umiejętności profesjonalnego redaktora. Pod wieloma względami praca Andy'ego jest równie ważna jak autorów. To samo dotyczy również redaktorów technicznych, którzy nadali książce obecny kształt. Wszystkie te poprawki zostały również wprowadzone w wersji elektronicznej, a więc w zawartości nie ma różnic.

Jednak wciąż wersja wydana przez O'Reilly'ego *będzie* inna. Jest porządnie oprawiona. Możesz mieć problemy z ładnym wydrukowaniem wersji domowej. Jest też

---

* Zwróć uwagę, że choć możesz wydrukować wersję elektroniczną, *nie* możesz kserować książki O'Reilly'ego ani sprzedawać żadnych jej kopii.

mało prawdopodobne, abyś uzyskał zbliżoną jakość, a jeśli już – to zapewne za dużo większe pieniądze. Ponadto nasze amatorskie ilustracje zostały w wersji drukowanej zastąpione innymi grafikami, pięknie przygotowanymi przez profesjonalnych artystów z wydawnictwa O' Reilly. Dla wersji drukowanej przygotowano też nowy, dokładniejszy indeks, dzięki czemu dużo łatwiej wyszukuje się informacje. Jeżeli ta książka jest czymś, co zamierzasz przeczytać od początku do końca, powinieneś zastanowić się nad przeczytaniem oficjalnej wersji drukowanej.

Przegląd treści

Rozdział 1, *Wprowadzenie do sieci*, omawia historię Linuksa i podaje podstawowe informacje o UUCP, TCP/IP, różnych protokołach, sprzęcie i bezpieczeństwie. Kolejne kilka rozdziałów omawia konfigurowanie Linuksa w sieci TCP/IP i uruchamianie podstawowych aplikacji. Nieco dokładniej przyglądamy się IP w rozdziale 2, *Wybrane problemy sieci TCP/IP*, za nim przejdziemy do edycji plików i tym podobnych tematów. Jeżeli wiesz już, jak działa ruting IP i na czym polega rozwiązywanie adresów, możesz pominąć ten rozdział.

Rozdział 3, *Konfigurowanie sprzętu sieciowego*, omawia podstawowe zagadnienia konfiguracyjne, takie jak tworzenie jądra i konfigurowanie karty Ethernet. Konfiguracja portów szeregowych jest przedstawiona oddzielnie w rozdziale 4, *Konfigurowanie urządzeń szeregowych*, ponieważ ten temat nie dotyczy jedynie sieci TCP/IP, ale ma także związek z UUCP.

Rozdział 5, *Konfigurowanie sieci TCP/IP*, pomaga skonfigurować maszynę w sieci TCP/IP. Zawiera wskazówki instalacyjne dla samodzielnych hostów z włączonym jedynie interfejsem pętli zwrotnej i hostów podłączonych do sieci Ethernet. Pokazuje także kilka przydatnych narzędzi, których możesz używać do testowania i debugowania swojej konfiguracji. Rozdział 6, *Usługi nazewnicze i konfigurowanie resolvera*, wyjaśnia, jak skonfigurować rozwiązywanie nazw i uruchomić serwer nazw.

Rozdział 7, *IP łącza szeregowego*, pokazuje, jak zestawić połączenie SLIP i szczegółowo omawia *dip* – narzędzie pozwalające na automatyzację większości niezbędnych kroków. Rozdział 8, *Protokół punkt-punkt*, jest poświęcony PPP i *pppd* – demonowi PPP.

Rozdział 9, *Firewall TCP/IP*, rozwija zagadnienia bezpieczeństwa sieciowego i opisuje firewall TCP/IP dla Linuksa oraz narzędzia do jego konfiguracji: *ipfwadm*, *ipchains* i *iptables*. Firewall IP zapewnia dokładną kontrolę nad tym, kto dostaje się do sieci i z jakiego hosta.

Rozdział 10, *Liczenie ruchu IP*, wyjaśnia, jak skonfigurować funkcję liczenia ruchu IP w Linuksie, tak aby śledzić, jak duży jest ruch wychodzący i kto go generuje.

Rozdział 11, *Maskowanie IP i translacja adresów sieciowych*, omawia własności specjalnego typu oprogramowania sieciowego Linuksa zwanego maskowaniem IP, które pozwala łączyć ze sobą całe sieci IP i korzystać z Internetu tylko przy użyciu jednego adresu IP, tak że wewnętrzna struktura sieci staje się niewidoczna.

Rozdział 12, *Ważne funkcje sieciowe*, stanowi krótkie wprowadzenie do konfigurowania pewnych ważniejszych aplikacji sieciowych, takich jak *rlogin*, *ssh* i tym podobne. Rozdział ten omawia również zarządzanie usługami przez *inetd* i podpowiada, w jaki sposób można zwiększyć bezpieczeństwo pewnych usług skierowanych do zaufanych hostów.

Rozdział 13, *System informacji sieciowej*, i rozdział 14, *Sieciowy system plików*, omawiają NIS i NFS. NIS to narzędzie używane do dystrybuowania informacji administracyjnych, takich jak hasła użytkownika w sieci lokalnej. NFS pozwala na współdzielenie systemów plików pomiędzy hostami w sieci.

W rozdziale 15, *IPX i system plików NCP*, omawiamy protokół IPX i system plików NCP. Pozwalają one zintegrować Linuksa ze środowiskiem Novell Netware przez współdzielenie plików oraz drukarek z maszynami nielinuksowymi.

Rozdział 16, *Zarządzanie UUCP Taylora*, stanowi wyczerpujące wprowadzenie do administrowania UUCP Taylora – darmową implementacją UUCP.

Pozostałe rozdziały książki szczegółowo przedstawiają pocztę elektroniczną i grupy dyskusyjne Usenetu. Rozdział 17, *Poczta elektroniczna*, wprowadza w główne zagadnienia poczty elektronicznej, takie jak wygląd adresów pocztowych i sposób, w jaki system obsługuje pocztę, by dotarła do adresata.

Rozdział 18, *Sendmail*, i rozdział 19, *Exim*, omawiają konfigurację programów *sendmail* i *exim* – dwóch małych agentów transportowych, które możesz wykorzystać w Linuksie. Przedstawiamy oba, ponieważ *exim* jest łatwiejszy do zainstalowania dla początkującego, a *sendmail* obsługuje UUCP.

Od rozdziału 20, *Grupy dyskusyjne*, do rozdziału 23, *Internet News*, wyjaśniamy obsługę wiadomości Usenetu i sposób instalacji i używania C News, *nntpd* i INN – trzech popularnych pakietów oprogramowania do zarządzania wiadomościami Usenetu. Po krótkim wprowadzeniu w rozdziale 20, możesz przeczytać rozdział 21, *C News*, jeżeli chcesz przesyłać wiadomości za pomocą C News – tradycyjnej usługi używanej wraz z UUCP. Kolejne rozdziały omawiają nowocześniejsze metody wykorzystujące protokół internetowy NNTP (*Network News Transport Protocol*). Rozdział 22, *NNTP i demon nntpd*, przedstawia sposób konfiguracji prostego demona NNTP o nazwie *nntp*, który zapewnia dostęp do czytania wiadomości w sieci lokalnej, zaś rozdział 23 opisuje silniejszy serwer do bardziej intensywnych transferów NetNews'ów: INN (*InterNet News*). I na koniec rozdział 24, *Konfigurowanie przeglądarki grup dyskusyjnych*, pokazuje, jak skonfigurować różne programy do czytania grup.

Konwencje zastosowane w tej książce

We wszystkich przykładach przedstawionych w tej książce zakładamy, że używasz powłoki, która jest kompatybilna z *sh*. Standardową powłoką wszystkich dystrybucji Linuksa jest *bash* kompatybilna z *sh*. Jeżeli korzystasz z *csh*, będziesz musiał odpowiednio zmodyfikować przykłady.

Poniżej przedstawiamy listę konwencji typograficznych użytych w książce:

Czcionka pochyła
: Używana do oznaczenia nazw plików i katalogów, programów i poleceń, opcji wiersza poleceń, adresów e-mail i ścieżki, URL i do podkreślenia nowych pojęć.

Czcionka pogrubiona
: Używana do nazw maszyn, hostów, ośrodków, użytkowników i ID oraz, okazjonalnie, do podkreślania pojęć.

`Czcionka o sta□ej szeroko□ci`
: Używana w przykładach do pokazania zawartości kodu plików lub wyniku działania poleceń oraz wskazywania zmiennych środowiskowych i słów kluczowych, które pojawiają się w kodzie.

`Czcionka pochy□a o sta□ej szeroko□ci`
: Używana do wskazania opcji zmiennych, słów kluczowych albo tekstu, który użytkownik ma zastąpić konkretną wartością.

`Czcionka pogrubiona o sta□ej szeroko□ci`
: Używana w przykładach do pokazania poleceń lub innego tekstu, który powinien być wpisywany przez użytkownika dosłownie.



Ramka z tą ikoną zawiera ostrzeżenie. Łatwo tu o błąd, który może źle się skończyć dla twojego systemu lub jest trudny od naprawienia.



Ramka z tą ikoną zawiera komentarz do pobliskiego tekstu.

Zgłaszanie uwag

Informacje zawarte w tej książce sprawdzaliśmy i weryfikowaliśmy na tyle, na ile byliśmy w stanie, ale pewne rzeczy mogły się zmienić (lub mogliśmy popełnić błąd!). Będziemy wdzięczni za powiadomienie nas o wszelkich dostrzeżonych błędach oraz podzielenie się swoimi sugestiami co do przyszłych wydań. Prosimy pisać na adres:

O'Reilly & Associates, Inc.
101 Morris Street
Sebastopol, CA 95472
1-800-998-9938 (w USA lub Kanadzie)
1-707-829-0515 (międzynarodowy lub lokalny)
1-707-829-0104 (faks)

Możesz nam także wysyłać wiadomości elektronicznie. Aby zapisać się na listę dyskusyjną lub poprosić o katalog, wyślij e-mail na adres:

info@oreilly.com

Aby poprosić o pomoc techniczną lub komentarz na temat książki, wyślij e-mail na adres:

bookquestions@oreilly.com

Prowadzimy witrynę WWW dla niniejszej książki. Znajdują się na niej przykłady, errata i plany przyszłych wydań. Strona ta znajduje się pod adresem:

http://www.oreilly.com/catalog/linag2

Więcej informacji na temat tej i innych książek znajdziesz w witrynie WWW wydawnictwa O'Reilly:

http://www.oreilly.com/

Podziękowania

To wydanie *Podręcznika administratora sieci* jest niemal wyłączną zasługą Olafa i Vince'a. Trudno docenić wysiłek włożony w badania i napisanie tego typu książki, jeżeli nie zrobi się tego samemu. Uaktualnianie książki było wyzwaniem – poważnym, ale dzięki dobrej podstawie także przyjemnym.

Książka wiele zawdzięcza tym, którzy poświęcili czas na jej korektę i pomogli usunąć wiele błędów, zarówno technicznych, jak i językowych. W tym działaniu znakomicie się uzupełniali Phil Hughes, John MacDonald i Erik Ratcliffe.

Serdeczne podziękowania kierujemy do członków zespołu redakcyjnego wydawnictwa O'Reilly, z którymi mieliśmy przyjemność pracować. Dziękujemy Sarah Jane Shangraw, która nadała książce obecny kształt; Maureen Dempsey, która redagowała tekst; Robowi Romano, Rhonowi Porterowi i Chrisowi Reilleyowi, którzy wykonali rysunki; Hannie Dyer, która zaprojektowała okładkę, Alicii Cech, Davidowi Futato i Jennifer Niedherst za układ wewnętrzny, Larsowi Kaufmanowi, który wpadł na pomysł zamieszczenia drzeworytów; Judy Hoer za indeks i na koniec Timowi O'Reilly'emu za odwagę podjęcia takiego projektu.

Na naszą wdzięczność zasłużyli też Andres Sepúlveda, Wolfgang Michaelis Michael K. Johnson i wszyscy programiści, którzy poświęcili wolny czas na sprawdzenie informacji zawartych w *Podręczniku administratora sieci*. Phil Hughes, John MacDonald i Eric Ratcliffe zgłosili nieocenione komentarze do drugiego wydania. Chcemy również podziękować wszystkim, którzy przeczytali pierwsze wydanie *książki* i przysłali poprawki i sugestie. Pełną, miejmy nadzieję, listę tych osób możesz znaleźć w pliku *Thanks* w wersji elektronicznej. Ostatecznie ta książka nie powstałaby bez wsparcia Holgera Grothego, który udostępnił Olafowi podłączenie do Internetu, niezbędne do powstania oryginalnej wersji.

Olaf chciałby również podziękować następującym grupom i firmom, które wydrukowały pierwsze wydanie *Podręcznika administratora sieci* i wsparły finansowo zarówno jego osobę, jak i cały Projekt Dokumentacji Linuksa: Linux Support Team, Er-

langen, Niemcy; S.u.S.E. GmbH, Fuerth, Niemcy; Linux System Labs, Inc., Clinton Twp., USA; Red Hat Software, Północna Karolina, USA.

Terry dziękuje swojej żonie Maggie, która niestrudzenie wspierała go w pracy na rzecz Projektu mimo wyzwania jakie stawiało przed nią urodzenie ich pierwszego dziecka, Jacka. Ponadto dziękuje *wielu* osobom ze społeczności Linuksa, dzięki którym osiągnął poziom pozwalający mu na wzięcie udziału w tym przedsięwzięciu. „Pomogę ci, jeżeli obiecasz pomóc za to komuś innemu”.

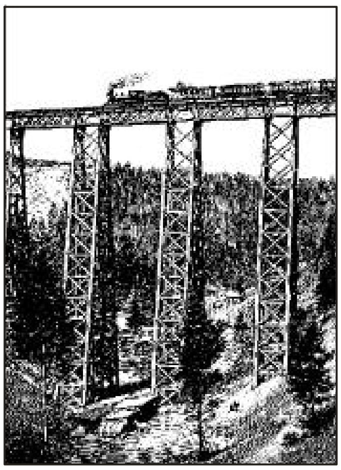
Jest jeszcze wiele osób, poza już wspomnianymi, które przyczyniły się do powstania *Podręcznika administratora sieci*. Zapoznali się z nim i przesyłali nam poprawki i sugestie. Jesteśmy im bardzo wdzięczni.

Oto lista tych, których działalność pozostawiła ślad w naszych folderach pocztowych.

Al Longyear, Alan Cox, Andres Sepúlveda, Ben Cooper, Cameron Spitzer, Colin McCormack, D.J. Roberts, Emilio Lopes, Fred N. van Kempen, Gert Doering, Greg Hankins, Heiko Eissfeldt, J.P. Szikora, Johannes Stille, Karl Eichwalder, Les Johnson, Ludger Kunz, Marc van Diest, Michael K. Johnson, Michael Nebel, Michael Wing, Mitch D' Souza, Paul Gortmaker, Peter Brouwer, Peter Eriksson, Phil Hughes, Raul Deluth Miller, Rich Braun, Rick Sladkey, Ronald Aarts, Swen Thüemmler, Terry Dawson, Thomas Quinot i Yury Shevchuk.

1

Wprowadzenie do sieci



Historia

Idea sie ci jest praw dop odo bnie tak sta ra jak sa ma ko mun ika cja. Się gnijmy do epoki ka mienn ej, kie dy to lu dzie używ ali bębn ów do prze syłania wia dom ości. Załóżmy, że ja skin iowi ec A chce za pros ić ja skin iowca B do gry w rzu can ie ka mien iami, ale miesz kają oni zbyt da leko od sie bie, by B usłyszał uder zenia A w bę ben. Co może zro bić ja skin iowi ec A? Może on 1) iść do miej sca za mieszk ania B, 2) użyć wię ks zego bęb na lub 3) po pros ić C, kt óry miesz ka w połowie dro gi po międ zy ni mi, aby przekazał ko mun ikat. Ostatnią mo żliw ość mo żna na zwać *siecią*.

Oczy wiś cie od czasów na szych pr zodków zmie niły się me to dy i urządze nia służące ko mu ni ka cji. Obec nie ma my kom pu te ry połączo ne ze sobą zwo ja mi dru tów, światłowo da mi, mi kro fa la mi i tym po dob ny mi; za ich po mocą uma wia my się na so bot ni mecz piłki no żnej*. Po niż ej opi sze my, ja ki mi środ ka mi i me to da mi mo żna nakłonić kom pu te ry do po ro zu mie wa nia się, choć po mi nie my i dru ty, i piłkę no żną.

W tym przewodniku opiszemy trzy typy sieci. Głównie skupimy się na sieciach opar tych na TCP/IP, który jest naj po pu lar niej szym ze sta wem pro to kołów sto so wanym zar ów no w sie ciach lo kal nych (*Lo cal Area Ne tworks* – LAN), jak i w sie ciach rozległych (*Wide Area Ne tworks* – WAN), takich jak Internet. Przyjrzymy się również pro to kołom UUCP i IPX. Swe go cza su UUCP był po wszech nie uży wa ny do przesyłania wiadomości Usenet i poczty przez komutowane połączenia telefoniczne. Obecnie jest mniej po pu lar ny, ale wciąż by wa przy dat ny w pew nych sy tu acjach. Protokół IPX jest uży wa ny prze wa żnie w śro do wi sku Novell Ne tWa re. Opi sze my, jak wy ko rzy stać go do podłącze nia ma szy ny linuk so wej do sie ci Novell. Ka żdy z wymienionych protokołów jest protokołem sieciowym służącym do przesyłania da-

* Co się jesz cze cza sem zda rza w Eu ro pie.

nych pomiędzy komputerami. Omówimy, jak są one używane, i pokażemy rządzące nimi zasady.

Sieć definiujemy jako zbiór *hostów*, które są w stanie komunikować się ze sobą, często za pośrednictwem pewnych wybranych spośród nich hostów, które rozsyłają dane pomiędzy uczestników. Hosty to często komputery, ale nie zawsze – za hosty można uznać także X terminale czy inteligentne drukarki. Niewielkie zbiorowiska hostów są nazywane również *ośrodkami* (ang. *sites*).

Komunikacja nie jest możliwa bez pewnego rodzaju języka czy kodu. W sieciach komputerowych te języki są nazywane *protokołami*. Jednak protokołu sieciowego nie powinieneś kojarzyć z pisemnym sprawozdaniem z zebrania. Trafniejsza jest analogia do sformalizowanych reguł zachowania obowiązujących, gdy na przykład spotykają się głowy państw, czyli do protokołu dyplomatycznego. Podobne protokoły używane w sieciach komputerowych to po prostu sztywne zasady wymiany komunikatów pomiędzy dwoma lub więcej hostami.

Sieci TCP/IP

Nowoczesne aplikacje sieciowe wymagają wyrafinowanego podejścia do przesyłania danych z jednej maszyny do drugiej. Jeżeli zarządzasz maszyną z Linuksem, z której korzysta wielu użytkowników, to może się zdarzyć, że wszyscy jednocześnie będą chcieli połączyć się ze zdalnymi hostami w sieci. Potrzebujesz więc sposobu, który pozwoli im współdzielić połączenie sieciowe bez przeszkadzania sobie wzajemnie. Rozwiązanie, które wykorzystuje wiele współczesnych protokołów sieciowych, nazywane jest *przełączaniem pakietów*. *Pakiet* to mała porcja danych, przesyłana przez sieć z jednej maszyny do drugiej. Przełączanie występuje w momencie, gdy datagram jest przenoszony przez dowolne łącze w sieci. W sieci z przełączaniem pakietów jedno łącze jest współdzielone przez wielu użytkowników w ten sposób, że przez to łącze pakiety są wysyłane kolejno od jednego użytkownika do drugiego.

Rozwiązanie, które przyjęło się w wielu systemach Unix, a następnie także w systemach nieuniksowych, nosi nazwę TCP/IP. Przy omawianiu sieci TCP/IP spotkasz się z określeniem *datagram*, które jest często używane wymiennie z określeniem *pakiet*, choć ma też inne, techniczne znaczenie. W tym podrozdziale przyjrzymy się podstawowym pojęciom związanym z TCP/IP.

Wprowadzenie do sieci TCP/IP

Początki TCP/IP sięgają programu badawczego finansowanego przez amerykańską agencję rządową DARPA (*Defense Advanced Research Projects Agency*) w 1969 roku. ARPANET była siecią eksperymentalną, która w 1975 roku, po latach zakończonych sukcesem badań, stała się siecią operacyjną.

W 1983 roku jako standard przyjęto nowy zestaw protokołów o nazwie TCP/IP, którego miały używać wszystkie hosty w sieci. Ostatecznie ARPANET przekształcił się w Internet (sam ARPANET przestał istnieć w 1990 roku), a zestaw TCP/IP jest stosowany także poza nim. Wiele firm stworzyło korporacyjne sieci TCP/IP, a Inter-

net osiągnął poziom, w którym można go uznać za wszechobecną technologię. Trudno jest, czytając gazetę lub czasopismo, nie zauważyć odnośników do Internetu – prawie każdy ma dziś do niego dostęp.

Aby nasze rozważania o TCP/IP oprzeć na czymś konkretnym, weźmy jako przykład sieć uniwersytetu Groucho Marx (GMU), znajdującego się gdzieś w Fredland. Większość wydziałów tej uczelni posiada własne sieci lokalne, jednak niektóre współdzielą jedną sieć, a inne mają ich po kilka. Wszystkie one są połączone ze sobą i podłączone do Internetu poprzez jedno szybkie łącze.

Załóżmy, że twój linuksowy komputer jest podłączony do sieci LAN zbudowanej z hostów uniksowych na wydziale matematyki i nazywa się **erdos**. Aby dostać się do hosta, powiedzmy **quark,** na wydziale fizyki, wprowadzasz następujące polecenie:

```
$ rlogin quark.physics
Welcome to the Physics Department at GMU
(ttyq2) login:
```

Po monicie wpisujesz nazwę użytkownika, powiedzmy **andres**, i swoje hasło. Następnie uzyskujesz dostęp do powłoki* komputera **quark**, w której możesz pisać tak, jakbyś siedział przy jego konsoli. Gdy wyjdziesz z powłoki, powracasz do monitu własnej maszyny. Właśnie użyłeś jednej z natychmiastowych, interaktywnych aplikacji, udostępnianych przez TCP/IP: zdalnego logowania.

Gdy jesteś zalogowany do maszyny **quark**, możesz również uruchomić aplikację graficzną, np. program procesora tekstów, program do rysowania czy przeglądarkę WWW. System X Window jest w pełni sieciowym środowiskiem graficznym, dostępnym dla wielu różnych systemów komputerowych. Aby powiedzieć aplikacji, że chcesz, aby na ekranie twojego hosta ukazywały się jej okna, musisz ustawić zmienną środowiskową `DISPLAY`:

```
$ DISPLAY=erdos.maths:0.0
$ export DISPLAY
```

Jeżeli teraz uruchomisz swoją aplikację, skontaktuje się ona z twoim X serwerem, a nie z tym działającym na **quarku**, i wyświetli wszystkie okna na twoim ekranie. Oczywiście na **erdosie** musi działać X11. Istota sprawy polega na tym, że TCP/IP pozwala **quarkowi** i **erdosowi** na wysyłanie pakietów X11 w tę i z powrotem, stąd masz wrażenie, że znajdujesz się w jednym systemie. Sieć jest tu niemal przezroczysta.

Kolejną bardzo ważną aplikacją TCP/IP jest NFS. Jej nazwa to skrót od słów *Network File System* (sieciowy system plików). Jest to inny sposób na spowodowanie, by sieć była przezroczysta. NFS pozwala na traktowanie hierarchii katalogów z innych hostów tak, jakby były one lokalnymi systemami plików, i sprawia, że wyglądają one jak inne katalogi na twoim hoście. Na przykład katalogi domowe wszystkich użytkowników mogą być przechowywane na serwerze centralnym, z którego mogą je montować wszystkie hosty w sieci LAN. W efekcie użytkownicy mogą logować się do dowolnej maszyny i znaleźć się w tym samym katalogu. Podobnie możliwe jest współdzielenie dużej liczby danych (takich jak bazy danych, dokumentacje czy apli-

* Powłoka to interfejs wiersza poleceń systemu operacyjnego Unix. Jest ona podobna do monitu DOS-a w środowisku Microsoft Windows, choć ma dużo większe możliwości.

kacje) przez wiele hostów w ten sposób, że na serwerze jest utrzymywana jedna baza danych, do której mają dostęp inne hosty. Do NFS-u powrócimy w rozdziale 14, *Sieciowy system plików*.

Oczywiście są to tylko przykłady tego, co możesz zrobić w sieciach TCP/IP. Możliwości są prawie nieograniczone i podczas lektury tej książki poznasz ich więcej.

Teraz przyjrzymy się bliżej sposobowi działania TCP/IP. Wiedza ta pomoże ci zrozumieć, jak musisz skonfigurować swój komputer i dlaczego. Rozpoczniemy od analizy sprzętu.

Ethernet

Najpopularniejszym rodzajem sprzętu w sieci lokalnej jest *Ethernet*. W najprostszej postaci składa się z jednego kabla i hostów podłączonych do niego przez wtyczki lub transceivery. Prosta instalacja ethernetowa jest stosunkowo niedroga, co wraz z przepustowością sieci rzędu 10, 100 czy nawet 1000 megabitów na sekundę przyczyniło się do dużej popularności tego standardu sprzętowego.

Ethernet występuje w trzech odmianach: *cienki*, *gruby* i *skrętkowy*. Cienki Ethernet i gruby Ethernet wykorzystują kable współosiowe, które różnią się średnicą i sposobem podłączania kabla do hosta. Cienki Ethernet wykorzystuje złącza „BNC" w kształcie litery T, które wkładasz w kabel i wkręcasz do gniazda z tyłu komputera. Gruby Ethernet wymaga wywiercenia niewielkiej dziurki w kablu i podłączenia transceivera za pomocą „zaczepu wampirowego" (ang. *vampire tap*). Następnie do transceivera można podłączyć hosty (jeden lub więcej). Cienki kabel ethernetowy może mieć maksymalnie 200 metrów długości, zaś kabel gruby – 500; ich nazwy to, odpowiednio, 10base-2 i 10base-5. „Base" odnosi się do modulacji pasma podstawowego (ang. *baseband modulation*) i po prostu oznacza, że dane są wysyłane do kabla bez pośrednio, bez żadnego modemu. Liczba na początku oznacza prędkość w megabitach na sekundę, a liczba na końcu maksymalną długość kabla w setkach metrów. Sieć skrętkowa wykorzystuje kabel zbudowany z dwóch par drutów miedzianych i zwykle wymaga dodatkowego urządzenia zwanego *hubem aktywnym*. Sieć skrętkowa jest także znana pod nazwą 10base-T, gdzie „T" oznacza skrętkę. Wersja sieci działająca z prędkością 100 megabitów nosi nazwę 100base-T.

Aby dodać host do sieci zbudowanej w oparciu o cienki Ethernet, musisz przerwać jej działanie na co najmniej kilka minut, ponieważ trzeba rozłączyć kabel i dołożyć wtyczki. Chociaż dodanie hosta do instalacji zbudowanej w oparciu o gruby Ethernet jest nieco bardziej skomplikowane, zwykle nie wymaga wyłączenia sieci. Ethernet oparty na skrętce jest jeszcze mniej kłopotliwy. Wykorzystuje urządzenie zwane hubem. Pełni ono rolę punktu podłączeniowego. Możesz dołączać hosty do huba lub odłączać je bez przerywania pracy całej sieci.

Wiele osób woli cienki Ethernet w małych sieciach, ponieważ jest on niedrogi. Karty PC kosztują około 30 USD (obecnie wiele firm dosłownie je wyrzuca), a kabel – kilka centów za metr. Jednak w dużych instalacjach lepszy jest gruby Ethernet lub skrętka. Na przykład na wydziale matematyki GMU pierwotnie wybrano gruby Ethernet, ponieważ kabel musiał być długi, a więc ruch nie może być zakłócany za każdym ra-

zem, gdy do sieci jest dodawany nowy host. Instalacje skrętkowe są obecnie bardzo popularne. Hubytanieją, a mniejsze jednostki można dostać za cenę, która jest atrakcyjna nawet dla małych sieci domowych. Okablowanie skrętkowe może być znacznie tańsze w przypadku dużych instalacji, a sam kabel jest dużo bardziej elastyczny niż kable współosiowe używane w innych rodzajach sieci Ethernet. Administratorzy sieci na wydziale matematyki GMU planują w przyszłym roku finansowym wymienić istniejące okablowanie i urządzenia na skrętkowe, by unowocześnić sieć i zaoszczędzić czas przy instalowaniu nowych hostów i przenoszeniu istniejących z miejsca na miejsce.

Jedną z wad technologii Ethernet jest ograniczenie długości kabla, co uniemożliwia jej zastosowanie w sieciach innych niż LAN. Jednak za pomocą wzmacniaków (ang. *repeater*), brydży i ruterów możliwe jest łączenie ze sobą segmentów sieci Ethernet. Wzmacniaki po prostu kopiują sygnały pomiędzy dwoma lub więcej segmentami tak, że wszystkie segmenty działają jak by to była jedna sieć Ethernet. Ze względu na wymagania czasowe, można umieścić co najwyżej cztery wzmacniaki pomiędzy dwoma hostami w sieci. Brydże i rutery są bardziej inteligentne. Analizują nadchodzące dane i przekazują je tylko wtedy, jeżeli docelowy host nie znajduje się w sieci lokalnej.

Ethernet działa na zasadzie systemu magistralowego, gdzie host może wysyłać pakiety (lub *ramki*) o wielkości do 1500 bajtów do innego hosta w tej samej sieci Ethernet. Host jest identyfikowany za pomocą sześciobajtowego adresu trwale zapisanego w oprogramowaniu firmowym interfejsu karty sieciowej Ethernet (*Network Interface Card*, NIC). Adresy te są zwykle sekwencją dwucyfrowych liczb szesnastkowych oddzielonych dwukropkami, czyli na przykład **aa:bb:cc:dd:ee:ff**.

Ramkę wysyłaną przez jedną stację widzą wszystkie podłączone stacje, ale tylko host, dla którego jest przeznaczona, odczytuje ją i przetwarza. Jeżeli dwie stacje próbują wysłać ramkę w tym samym czasie, dochodzi do *kolizji*. Kolizje w sieci Ethernet są wykrywane bardzo szybko przez elektronikę kart interfejsu i są rozwiązywane przez przerwanie wysyłania z obu stacji, odczekanie przez każdą z nich losowego przedziału czasu i ponowną próbę transmisji. Nieraz spotkasz się z opinią, że kolizje w Ethernecie są problemem i że przez nie wykorzystanie Ethernetu wynosi zaledwie około 30 procent dostępnego pasma. Kolizje są zjawiskiem typowym dla sieci Ethernet i nie powinien nieś być zaskoczony, zwłaszcza jeśli sieć jest przeciążona. Może ich być maksymalnie 30 procent. Wykorzystanie sieci Ethernet jest w rzeczywistości ograniczone do około 60 procent – dopiero jeżeli nie osiągniesz tej wartości, to możesz zacząć się martwić*.

Inne typy urządzeń

W większych instalacjach, takich jak na uniwersytecie Groucho Marx, Ethernet zwykle nie jest jedynym typem używanego sprzętu. Istnieje wiele innych protokołów

* Lista pytań FAQ dotycząca Ethernetu, która znajduje się pod adresem *http://www.faqs.org/faqs/LANs/ethernet-faq/*, omawia to zagadnienie, a spory zasób szczegółowych informacji historycznych i technicznych jest dostępny na stronie poświęconej Ethernetowi prowadzonej przez Charlesa Spurgeona pod adresem *http://wwwhost.ots.utexas.edu/ethernet/*.

przesyłania danych, które można wykorzystywać. Wszystkie wymienione poniżej protokoły są obsługiwane przez Linuksa, ale ze względu na ograniczoną ilość miejsca przedstawimy je skrótowo. Szczegółowy opis wielu innych protokołów znajduje się w odpowiednich dokumentach HOWTO, możesz tam zajrzeć, jeżeli jesteś zainteresowany poznaniem tych, których nie opisujemy w naszej książce.

Na uniwersytecie Groucho Marx sieć LAN każdego wydziału jest podłączona do szybkiej sieci szkieletowej, w której wykorzystano światłowód i technologię sieciową FDDI (*Fiber Distributed Data Interface*). FDDI prezentuje całkiem inne podejście do przesyłania danych, zasadniczo polegające na wysyłaniu *żetonów* (ang. *tokens*). Stacja ma prawo wysłać ramkę tylko wtedy, jeżeli wcześniej odbierze żeton. Główną zaletą protokołu przekazywania żetonów jest zmniejszenie liczby kolizji. Protokół może dużo prościej osiągnąć pełną prędkość przesyłania, w przypadku FDDI do 100 Mb/s. FDDI oparte na światłowodzie ma wiele zalet, ponieważ dopuszczalna długość kabla jest dużo większa niż w technologiach wykorzystujących zwykły kabel miedziany. Limit wynosi tutaj około 200 km, co sprawia, że FDDI znakomicie nadaje się do łączenia wielu budynków w mieście lub, tak jak w naszym przykładzie, wielu budynków campusu.

Podobnie jeżeli w okolicy znajdują się urządzenia sieciowe firmy IBM, prawdopodobnie zainstalowano sieć IBM Token Ring. Token Ring jest stosowana jako alternatywa dla Ethernetu w niektórych sieciach LAN i ma te same zalety co FDDI, jeśli chodzi o prędkość, ale mniejszą przepustowość (4 lub 16 Mb/s). Jest też tańsza, ponieważ wykorzystuje kabel miedziany, a nie światłowody. W Linuksie sieć Token Ring jest konfigurowana prawie tak samo jak Ethernet, a więc nie musimy jej tu taj poświęcać więcej uwagi.

W sieciach lokalnych LAN mogą być też stosowane inne technologie, takie jak ArcNet czy DECNet, choć obecnie już rzadziej spotyka się je. Linux również je obsługuje, ale nie będziemy ich tu opisywać.

Wiele sieci państwowych obsługiwanych przez firmy telekomunikacyjne wykorzystuje protokoły przełączania pakietów. Chyba największą popularnością cieszy się standard o nazwie X.25. Wiele sieci publicznych, takich jak Tymnet w USA, Austpac w Australii i Datex-P w Niemczech, oferuje tę usługę. X.25 definiuje zestaw protokołów sieciowych, które opisują, jak urządzenie będące terminalem danych, takie jak host, łączy się ze urządzeniem do przesyłania danych (przełącznikiem X.25). X.25 wymaga synchronicznego łącza danych, a zatem specjalnego synchronicznego portu szeregowego. Można stosować X.25 z normalnymi portami szeregowymi pod warunkiem, że ma się specjalne urządzenie o nazwie PAD (*Packet Assembler Disassembler*). PAD jest samodzielnym urządzeniem udostępniającym synchroniczne i asynchroniczne porty szeregowe. Obsługuje protokół X.25, tak więc proste urządzenia terminalowe mogą nawiązywać i przyjmować połączenia realizowane za pomocą tego protokołu. X.25 jest często używany do przesyłania innych protokołów sieciowych, takich jak TCP/IP. Ponieważ datagramy IP nie mogą być w prosty sposób przetłumaczone na X.25 (lub odwrotnie), są one enkapsulowane w pakietach X.25 i wysyłane przez sieć. W Linuksie dostępna jest eksperymentalna implementacja protokołu X.25.

Nowszym protokołem, powszechnie oferowanym przez firmy telekomunikacyjne, jest *Frame Relay*. Pod względem technicznym ma on wiele wspólnego z protokołem X.25, ale w działaniu bardziej przypomina protokół IP. Podobnie jak X.25, tak Frame Relay wymaga specjalnego synchronicznego portu szeregowego. Stąd wiele kart obsługuje oba te protokoły. Alternatywą jest urządzenie nazywane Frame Relay Access Device (FRAD) obsługujące enkapsulację pakietów Ethernet w pakietach Frame Relay na czas transmisji w sieci. Frame Relay znakomicie nadaje się do przesyłania pakietów TCP/IP pomiędzy ośrodkami. Linux jest wyposażony w sterowniki, które obsługują pewne typy wewnętrznych urządzeń Frame Relay.

Jeżeli potrzebujesz szybszej sieci, która będzie przesyłać wiele różnych i nietypowych rodzajów danych, takich jak cyfrowo zapisany głos i wideo, to zapewne zainteresuje cię ATM (*Asynchronous Transfer Mode*). ATM jest nową technologią sieciową, która została zaprojektowana tak, by zapewnić łatwe zarządzanie, dużą prędkość, małe opóźnienia przy przesyłaniu danych i kontrolę nad jakością usług (*Quality of Service* – QS). Wiele firm telekomunikacyjnych, które liczą na usprawnienie zarządzania siecią i jej obsługi, sięga po infrastrukturę sieci ATM, ponieważ pozwala ona łączyć wiele różnych usług sieciowych na jednej platformie. ATM jest często używana do przesyłania protokołu TCP/IP. W *Networking-HOWTO* znajdziesz informacje na temat obsługi ATM-u przez Linuksa.

Często radioamatorzy wykorzystują swój sprzęt do łączenia komputerów w sieć – powszechnie nosi to nazwę *radia pakietowego* (ang. *packet radio*). Jednym z protokołów wykorzystywanych przez nich jest AX.25, w pewnym stopniu oparty na X.25. Radioamatorzy używają protokołu AX.25 do przesyłania TCP/IP, a także innych protokołów. AX.25, podobnie jak X.25, wymaga urządzenia szeregowego, które może działać w trybie synchronicznym lub urządzenia zewnętrznego o nazwie Terminal Node Controller, które konwertuje pakiety przesyłane przez łącze asynchroniczne na pakiety przesyłane synchronicznie. Istnieje szereg różnych kart interfejsów obsługujących radiopakietowe – mówi się, że są to karty oparte na Z8530 SCC. Nazwa ta odnosi się do najpopularniejszego kontrolera komunikacyjnego używanego do ich budowy. Dwa inne protokoły, często przesyłane przez AX.25, to NetRom i Rose – są to protokoły warstwy sieciowej. Ponieważ protokoły te działają w oparciu o AX.25, mają te same wymagania sprzętowe. Linux w pełni implementuje protokoły AX.25, NetRom i Rose. *AX25-HOWTO* jest dobrym źródłem informacji na temat implementacji tych protokołów w Linuksie.

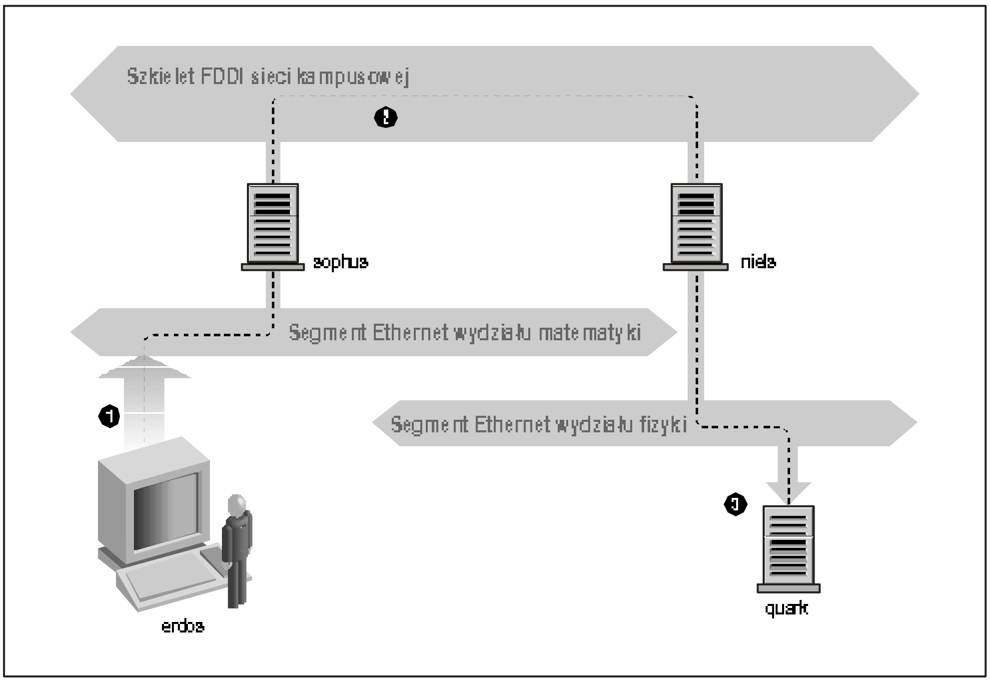
Dostęp do Internetu można również uzyskać poprzez połączenia komutowane do systemu centralnego korzystające z wolnych, ale tanich łączy szeregowych (telefon, ISDN i tym podobne). Wymagają one jeszcze innych protokołów przesyłania pakietów, takich jak SLIP czy PPP. Opiszemy je później.

Protokół internetowy (IP)

Zapewne szczytem twoich marzeń nie jest sieć oparta na jednym połączeniu ethernetowym lub typu punkt-punkt. Idealnie byłoby, gdybyś mógł łączyć się z hostem bez względu na to, jakiego typu łączem fizycznym jest podłączony do sieci. Na przykład w dużych instalacjach, takich jak na przykład owym uniwersytecie Groucho

Marx, zwykle masz kilka oddzielnych sieci, które są w pewien sposób ze sobą połączone. Wydział matematyki ma dwie sieci Ethernet, jedną z szybkimi maszynami dla profesorów i absolwentów, a drugą z wolnymi komputerami dla studentów. Obie są podłączone do szkieletowej sieci FDDI w campusie.

Połączenie to jest obsługiwane przez dedykowany host zwany *gatewayem*, który obsługuje nadchodzące i wychodzące pakiety, kopiując je między dwoma Ethernetami i kablem optycznym w sieci FDDI. Na przykład gdy byś był na wydziale matematyki i chciałbyś dostać się ze swojego Linuksa do komputera **quark** na wydziale fizyki, oprogramowanie sieciowe nie wysłałoby pakietów bezpośrednio do komputera **quark**, ponieważ nie znajduje się on w tym samym segmencie Ethernet. W tym wypadku jako przekaźnik zostanie wykorzystany gateway. Gateway (o nazwie **sophus**) przekazuje te pakiety poprzez sieć szkieletową do gatewaya **niels** na wydziale fizyki. Dopiero **niels** dostarcza je do docelowej maszyny. Przepływ danych pomiędzy komputerami **erdos** i **quark** pokazano na rysunku 1-1.



Rysunek 1-1. Trzy etapy wysyłania datagramu z erdosa do quarka

Taki sposób przekazywania danych do zdalnego hosta nazywa się *rutowaniem*, a w tym kontekście pakiety często są nazywane *datagramami*. Aby uprościć opisaną procedurę, wymianę datagramów niezależnie od stosowanych urządzeń, powierza się zawsze temu samemu protokołowi, który nosi nazwę *protokołu internetowego* (*Internet Protocol* – IP). W rozdziale 2, *Wybrane problemy sieci TCP/IP*, opiszemy bardziej szczegółowo zarówno IP, jak i ruting.

Główną zaletą IP jest to, że przekształca on fizycznie różne od siebie sieci w jedną, stuprocentowo homogeniczną sieć. Nazywa się to współdziałaniem międzysieciowym (ang. *internetworking*), a uzyskana „metasieć" to *internet*. Zwróć tutaj uwagę na subtelną różnicę pomiędzy nazwami „internet" a „Internet". Ta ostatnia to nazwa własna konkretnego internetu o globalnym zasięgu.

Oczywiście IP wymaga także niezależnego od sprzętu schematu adresowania. Taki schemat to unikalny 32-bitowy numer przypisywany każdemu hostowi, zwany *adresem IP*. Adres IP ma zwykle postać czterech liczb dziesiętnych, oddzielonych kropkami, po jednej liczbie na każdy 8-bitowy segment. Na przykład **quark** mógłby mieć adres IP o postaci **0x954C0C04**, który byłby zapisany jako **149.76.12.4**. Format ten jest również nazywany *kropkową notacją dziesiętną* (ang. *dotted decimal notation*), a czasem *kropkową notacją czwórkową* (ang. *dotted quad notation*). Dla adresów IP co raz częściej spotyka się nazwę IPv4 (od *Internet Protocol Version 4*), ponieważ nowy standard o nazwie IPv6 oferuje dużo bardziej elastyczny sposób adresowania oraz inne, nowoczesne własności. Ale minie co najmniej rok od wydania tej książki, zanim wejdzie on w życie.

Zapewne już zauważyłeś, że mamy teraz trzy różne typy adresów: pierwszy to nazwa hosta, jak **quark**, następnie adres IP i jeszcze adresy sprzętowe, takie jak 6-bajtowy adres ethernetowy. Wszystkie te adresy w pewien sposób muszą do siebie pasować, tak żeby po napisaniu *rlogin quark*, oprogramowanie sieciowe mogło podać adres IP komputera **quark**, a następnie znaleźć adres ethernetowy odpowiadający adresowi IP, wtedy gdy IP dostarczy już dane do sieci Ethernet na wydziale fizyki.

Sytuację tę omówimy w rozdziale 2. Teraz wystarczy zapamiętać, że wyszukiwanie adresów jest nazywane albo *rozwiązywaniem nazwy hosta*, jeśli dotyczy zamiany nazw hostów na adresy IP, albo *rozwiązywaniem adresów*, jeśli ma nastąpić zamiana tych drugich na adresy sprzętowe.

IP w łączach szeregowych

W łączach szeregowych obowiązuje standard znany jako SLIP (*Serial Line IP* – protokół internetowy łącza szeregowego). Zmodyfikowana wersja SLIP, znana pod nazwą CSLIP (*Compressed SLIP* - SLIP z kompresją), kompresuje nagłówki IP, co pozwala lepiej wykorzystać relatywnie małą przepustowość cechującą większość łączy szeregowych. Innym protokołem szeregowym jest PPP (*Point-to-Point Protocol* – protokół punkt-punkt). PPP jest bardziej nowoczesny, niż SLIP i posiada różne właściwości, które stanowią o jego większej atrakcyjności. Jego główną zaletą w stosunku do SLIP jest to, że nie ogranicza się do przesyłania datagramów IP, ale jest zaprojektowany tak, by można było przez niego przesyłać dowolny protokół.

Protokół kontroli transmisji (TCP)

Wysyłanie datagramów z jednego hosta do innego to nie wszystko. Jeżeli zalogujesz się do **quarka**, będziesz chciał mieć niezawodne połączenie pomiędzy twoim procesem *rlogin* na **erdosie** a procesem powłoki na **quarku**. Tak więc informacja wysłana tam i z powrotem musi być podzielona na pakiety przez nadawcę i ponownie

połączona w ciąg znaków przez odbiorcę. Choć wydaje się to trywialne, jest jednak czynnością dość złożoną.

Należy pamiętać, że IP jest z założenia protokołem zawodnym. Załóżmy, że dziesięć osób w twojej sieci Ethernet rozpoczęło pobieranie najnowszej wersji kodu źródłowego przeglądarki Netscape z serwera FTP należącego do przykładowego uniwersytetu. Wygenerowany w ten sposób ruch może być za duży dla gatewaya, który jest za wolny, i ma za mało pamięci. Jeżeli teraz zdarzy się, że wyślesz pakiet do **quarka**, **sophusowi** może zabraknąć przez chwilę miejsca na buforowanie i nie będzie w stanie przekazać twojego pakietu. W tej sytuacji IP po prostu gubi pakiet, który znika bezpowrotnie. Dlatego to komunikujące się hosty są odpowiedzialne za sprawdzenie integralności i kompletności danych oraz ich ponowną transmisję w przypadku błędu.

To zadanie jest realizowane przez inny protokół – TCP *(Transmission Control Protocol)*, który jest niezawodną usługą ponad IP. Istotną własnością TCP jest to, że używa on IP, by dać ci wrażenie prostego połączenia pomiędzy dwoma procesami, odpowiednio na twoim hoście i zdalnej maszynie, a więc nie musisz martwić się o to, jak i którędy są w rzeczywistości przesyłane twoje dane. Połączenie TCP działa w rzeczywistości jak dwukierunkowy potok, do którego oba procesy mogą zapisywać i odczytywać. Dobra jest tu analogia do rozmowy telefonicznej.

TCP identyfikuje punkty końcowe każdego połączenia po adresach IP dwóch komunikujących się hostów i numerach *portów* na każdym z nich. Porty mogą być postrzegane jako punkty zaczepienia połączeń sieciowych. Gdybyśmy wrócili do przykładu z rozmową telefoniczną, to można sobie wyobrazić, że hosty to miasta, zaś adresy IP to numery kierunkowe (gdzie numery odwzorowują miasta), a numery portów to konkretne numery lokalne (gdzie numery odwzorowują indywidualne numery telefonów). Pojedynczy host może realizować wiele różnych usług, rozpoznawanych po numerze portu.

W przykładzie *rlogin*, aplikacja klienta (*rlogin*) otwiera port na hoście **erdos** i łączy się z portem 513 hosta **quark**, na którym nasłuchuje serwer *rlogind*. W ten sposób zostaje nawiązane połączenie TCP. Za pomocą tego połączenia *rlogind* przeprowadza procedurę autoryzacji, a następnie uruchamia powłokę. Standardowe wejście i wyjście powłoki są przekierowywane na połączenie TCP, a więc wszystko, co napiszesz w aplikacji *rlogin* na swojej maszynie, zostanie przekazane przez strumień TCP i podane powłoce jako standardowe wejście.

Protokół datagramów użytkownika (UDP)

Oczywiście TCP nie jest jedynym protokołem użytkownika w sieci TCP/IP. Choć jest odpowiedni dla aplikacji takich jak *rlogin*, jego złożoność uniemożliwia wykorzystanie go w aplikacjach, takich jak NFS, które z kolei używają bratniego protokołu – UDP (*User Datagram Protocol* – protokół datagramów użytkownika). Podobnie jak TCP, UDP pozwala aplikacji na łączenie się z usługą na pewnym porcie zdalnej maszyny, ale bez zestawiania połączenia. Można go wykorzystać do wysyłania pojedynczych pakietów do docelowej usługi – stąd nazwa.

Załóżmy, że chcesz poprosić o niewielką porcję danych z serwera baz danych. Zestawienie połączenia TCP wymaga co najmniej trzech datagramów, kolejne trzy są potrzebne do wysłania i potwierdzenia niewielkiej porcji danych w każdą stronę, a następne trzy do zamknięcia połączenia. UDP obsłuży takie połączenie za pomocą tylko dwóch datagramów, a efekt końcowy będzie taki sam. UDP jest protokołem bezpołączeniowym i nie wymaga zestawiania i zamykania sesji. Po prostu umieszczamy dane w datagramie i wysyłamy go do serwera – serwer przygotowuje odpowiedź, umieszcza dane w datagramie zaadresowanym zwrotnie (do nas) i przesyła go z powrotem. Choć w przypadku prostych transakcji UDP działa szybciej i bardziej efektywnie niż TCP, nie reaguje na gubienie datagramów. Dbałość o kompletność danych pozostawia się aplikacji, na przykład aplikacji serwera nazw.

Więcej na temat portów

Porty można traktować jako punkty zaczepienia połączeń sieciowych. Jeżeli aplikacja chce udostępnić jakąś usługę, podłącza się sama do portu i czeka na klientów (często nazywa się to *nasłuchiwaniem* na porcie). Klient, który chce skorzystać z tej usługi, alokuje port na swoim hoście lokalnym i podłącza się do portu serwera na hoście zdalnym. Ten sam port może być otwarty na wielu różnych maszynach, ale na każdej maszynie tylko jeden proces może otworzyć port w danej chwili.

Istotną własnością portów jest to, że gdy zostanie ustanowione połączenie pomiędzy klientem a serwerem, inna kopia serwera może podłączyć się do portu serwera i oczekiwać kolejnych klientów. Ta właściwość pozwala na przykład na kilka jednoczesnych zdalnych logowań do tego samego hosta wykorzystujących port 513. TCP jest w stanie rozróżnić te połączenia, ponieważ przychodzą one z różnych portów lub hostów. Jeżeli zalogujesz się dwukrotnie z **erdosa** do **quarka**, pierwszy klient *rlogin* wykorzysta port lokalny 1023, a drugi port 1022. Jednak oba podłączą się do tego samego portu 513 hosta **quark**. Te dwa połączenia będą rozróżniane poprzez numery portów na **erdosie**.

W powyższym przykładzie użyto portów jako miejsca spotkania – klient kontaktuje się z określonym portem, by uzyskać daną usługę. Aby klient wiedział, z jakim numerem portu ma się kontaktować, administratorzy obu systemów muszą uzgodnić przypisanie numerów portów. W przypadku popularnych usług, takich jak *rlogin*, numerami tymi administruje centralnie organizacja IETF (*Internet Engineering Task Force*), która regularnie publikuje RFC o nazwie *Assigned Numbers* (RFC-1700). Dokument ten zawiera między innymi numery portów przypisane dobrze znanym usługom. Linux wykorzystuje plik o nazwie */etc/services*, który kojarzy nazwy usług z numerami portów.

Warto zauważyć, że choć zarówno połączenia TCP, jak i UDP opierają się na portach, to ich numery nie kłócą się ze sobą. Oznacza to, że na przykład port 513 TCP różni się od portu 513 UDP. W rzeczywistości porty te działają jako punkty dostępu dla dwóch różnych usług: *rlogin* (TCP) i *rwho* (UDP).

Biblioteka socket

W uniksowych systemach operacyjnych oprogramowanie realizujące wszystkie zadania i obsługujące opisane powyżej protokoły jest zwykle częścią jądra. Podobnie jest w Linuksie. Najpopularniejszym interfejsem programowania w świecie Uniksa jest biblioteka *Berkeley Socket*. Jej nazwa wywodzi się z popularnej analogii, w której port jest postrzegany jako gniazdo, a podłączanie się do portu – jako włączanie do gniazda. Biblioteka udostępnia wywołanie *bind*, w którym podaje się zdalny host, protokół transportowy i usługę, do której program może się podłączyć lub której ma nasłuchiwać (za pomocą *connect*, *listen* i *accept*). Biblioteka *socket* jest nieco bardziej ogólna, ponieważ udostępnia nie tylko klasę gniazd opartych na TCP/IP (gniazda *AF_INET*), ale także klasę, która obsługuje połączenia lokalne do maszyny (klasa *AF_UNIX*). Niektóre implementacje mogą także obsługiwać inne klasy, takie jak protokół XNS (*Xerox Networking System*) lub X.25.

W Linuksie biblioteka *socket* jest częścią standardowej biblioteki *lib C*. Obsługuje gniazda *AF_INET* i *AF_INET6* dla TCP/IP oraz *AF_UNIX* dla gniazd domeny Uniksa. Obsługuje również gniazda *AF_IPX* dla protokołów sieci Novell, *AF_X25* dla protokołu sieci X.25, *AF_ATMPVC* i *AF_ATMSVC* dla protokołów sieci ATM i *AF_AX25*, *AF_NETROM* i *AF_ROSE* dla protokołów radia amatorskiego. Inne rodziny protokołów są w trakcie tworzenia i będą stopniowo dodawane.

Sieci UUCP

UUCP (*Unix-to-Unix Copy Program* – program kopiujący między systemami uniksowymi) był pakietem programów, które przesyłały pliki po łączach szeregowych, rozplanowywały te przesłania w czasie i inicjowały wykonywanie programów w zdalnych ośrodkach. Od czasu pierwszej implementacji, pod koniec lat siedemdziesiątych, UUCP znacznie się zmieniło, chociaż zakres oferowanych usług pozostał niewielki. UUCP stosuje się głównie w sieciach rozległych (WAN), opartych o okresowo uruchamiane łącza komutowane.

UUCP stworzono w Bell Laboratories w 1977 roku w celu zapewnienia komunikacji pomiędzy ośrodkami programistycznymi pracującymi pod Uniksem. W połowie 1978 roku sieć łączyła już ponad 80 ośrodków. Działała w niej poczta elektroniczna oraz zdalne drukowanie. Jednak podstawowym zastosowaniem systemu była dystrybucja nowego oprogramowania i poprawianie błędów. Obecnie UUCP nie jest ograniczone wyłącznie do środowiska Unix. Istnieją darmowe i komercyjne wersje dla wielu innych platform, takich jak AmigaOS, DOS i Atari TOS.

Jedną z głównych wad sieci UUCP jest to, że działają one wsadowo. Zamiast stałego połączenia pomiędzy hostami, wykorzystują połączenia tymczasowe. Host UUCP może połączyć się z innym hostem UUCP tylko raz dziennie i to na krótko. W czasie trwania połączenia przesyła wszystkie grupy dyskusyjne, pocztę i pliki, które znajdują się w kolejce, a następnie się rozłącza. To właśnie konieczność kolejkowania ogranicza różnorodność zastosowań UUCP. W przypadku poczty elektronicznej, użytkownik może przygotować wiadomość e-mail i wysłać ją. Będzie ona oczekiwać w kolejce na hoście UUCP, aż zadzwoni on do innego hosta, by przesłać wiadomość.

Jest to do przyjęcia w przypadku usług sieciowych takich jak poczta elektroniczna, ale nie nadaje się dla innych usług, na przykład *rlogin*.

Pomimo tych ograniczeń, wciąż na świecie istnieje wiele sieci UUCP utrzymywanych głównie przez hobbystów, którzy oferują prywatnym użytkownikom dostęp do Internetu za rozsądną cenę. Głównym powodem długotrwałej popularności UUCP była jej atrakcyjność cenowa w porównaniu z bezpośrednim podłączeniem do Internetu. Aby zrobić z twojego komputera węzeł UUCP, potrzebujesz jedynie modemu, działającej implementacji UUCP i innego węzła UUCP, który będzie chciał przyjmować twoją pocztę i grupy dyskusyjne. Wiele osób chętnie obsługiwało ruch UUCP dla indywidualnych użytkowników, ponieważ takie połączenia nie zakłócały zbytnio pracy ich sieci.

Konfigurację UUCP omawiamy w jednym z dalszych rozdziałów książki, choć czynimy to skrótowo, gdyż protokół ten jest obecnie wypierany przez TCP/IP. Dostęp do Internetu jest powszechny i nie stanowi problemu w większości zakątków świata.

Sieć w Linuksie

Linux, który powstaje wspólnym wysiłkiem programistów z całego świata, nie byłby możliwy bez sieci globalnej. Nie ma więc nic dziwnego w tym, że od samego początku pracowano nad zapewnieniem mu zdolności sieciowych. Implementacja UUCP działała już w pierwszych wersjach Linuksa, a prace nad siecią opartą na TCP/IP rozpoczęły się jesienią 1992 roku, kiedy Ross Biro wraz z grupą programistów stworzyli to, co teraz jest znane pod nazwą Net-1.

Po Rossie, który odszedł w maju 1993 roku, pracę nad nową implementacją kontynuował Fred van Kempen, przepisując główne części kodu. Projekt ten był znany jako Net-2. Pierwsza publiczna wersja, Net-2d, została udostępniona w lecie 1993 roku (jako część jądra 0.99.10) i od tego czasu była utrzymywana i rozwijana przez kilka osób, a przede wszystkim przez Alana Coxa*. Oryginalne prace Alana były znane pod nazwą Net-2Debugged, gdyż uwolnił on kod od wielu błędów i wprowadził liczne udoskonalenia. Od wersji 1.0 Linuksa kod sieciowy Alana nosił nazwę Net-3. Kod ten był dalej rozwijany w Linuksie 1.2 i 2.0. Jądra 2.2 i nowsze wykorzystują wersję Net-4, która pozostaje standardem do chwili obecnej.

Kod sieciowy Linuksa Net-4 oferuje różnorodne sterowniki urządzeń i zaawansowane własności. Do standardowych protokołów Net-4 zaliczają się: SLIP i PPP (do przesyłania danych przez łącza szeregowe), PLIP (dla łączy równoległych), IPX (dla sieci kompatybilnych z Novellem, które omówimy w rozdziale 15, *IPX i system plików NCP*), Appletalk (dla sieci Apple) i AX.25, NetRom i Rose (dla sieci radioamatorskich). Inne standardy obsługiwane przez Net-4 to: firewall IP, liczenie ruchu IP (omawiane w rozdziałach 9 i 10) i maskowanie IP (omawiane w rozdziale 11, *Maskowanie IP i translacja adresów sieciowych*). Zaawansowane algorytmy rutingu i tunelowanie IP są obsługiwane na kilka możliwych sposobów. W Net-4 za warto sterowni-

* Do Alana można pisać na adres *alan@lxorguk.ukuu.org.uk*.

ki dla szeregu urządzeń Ethernet, a także dla FDDI, Token Ring, Frame Relay i ISDN oraz ATM.

Ponadto istnieje tu wiele innych właściwości, które znacznie rozszerzają elastyczność Linuksa. Należą do nich implementacja systemu plików SMB, która współdziała z takimi aplikacjami, jak *lanmanager* i Microsoft Windows, oraz implementacja Novell NCP (*NetWare Core Protocol*)*.

Różne ścieżki rozwoju

W różnych okresach w różnych kierunkach rozwijano oprogramowanie sieciowe dla Linuksa.

Po uznaniu Net-2Debugged za implementację sieci, Fred nadal pracował nad kodem sieciowym. W rezultacie powstała wersja kodu o nazwie Net-2e, która charakteryzowała się dużo lepiej przemyślaną konstrukcją warstwy sieciowej. Fred chciał też ustandaryzować interfejs sterowników urządzeń (*Device Driver Interface* – DDI), ale prace nad Net-2e zakończono.

Inna implementacja sieci TCP/IP pochodzi od Matthiasa Urlichsa, który napisał sterownik ISDN dla Linuksa i FreeBSD. W tym celu zintegrował on część kodu sieciowego BSD z jądrem Linuksa. Projekt ten również nie jest rozwijany.

Wiele się zmieniło w implementacji sieci w jądrze Linuksa, i wciąż się zmienia. Czasem oznacza to, że zmiany muszą wystąpić także w innym oprogramowaniu, takim jak narzędzia do konfiguracji sieci. Choć nie jest to obecnie tak dużym problemem jak niegdyś, jednak wciąż może się zdarzyć, że jeśli zainstalujesz nowszą wersję jądra, to narzędzia do konfiguracji sieci również będą wymagały uaktualnienia. Na szczęście w większości obecnych dystrybucji Linuksa jest to proste zadanie.

Implementacja sieci Net-4 jest produktem w pełni dopracowanym, stosowanym w bardzo wielu ośrodkach na całym świecie. Wiele wysiłku włożono w poprawę wydajności implementacji Net-4 i teraz może ona konkurować z najlepszymi implementacjami dostępnymi dla danych platform sprzętowych. Linux cieszy się coraz większym wzięciem wśród dostawców Internetu, gdzie często jest używany do tworzenia tanich i niezawodnych serwerów WWW, serwerów pocztowych i serwerów grup dyskusyjnych dla tego typu organizacji. Obecnie zainteresowanie rozwojem Linuksa jest na tyle duże, że wszystkie zmiany w technologii sieciowej znajdują swoje odzwierciedlenie w kolejnych wersjach jądra, a jego najnowsze wersje oferują jako standard kolejną generację protokołu IP IPv6.

Skąd wziąć kod

Dziś wydaje się dziwne, że w początkach rozwoju kodu sieciowego Linuksa standardowe jądro wymagało ogromnego pakietu poprawek dodające go obsługę sieci. Obecnie obsługa sieci jest uwzględniona w głównym jądrze Linuksa. Ostatnie stabilne jądra Linuksa można znaleźć w ośrodku **ftp.kernel.org** w katalogu */pub/linux/kernel/v2.x/*, gdzie *x* jest liczbą parzystą. Najnowsze eksperymentalne wersje jądra

* NCP jest protokołem, na którym oparte są system plików i usługi drukowania w Novellu.

Linuk sa mo żna zna le źć w ośrod ku **ftp.kernel.org** w ka ta lo gu */pub/linux/kernel/v2.y/*, gdzie *y* jest liczbą nie pa rzystą. Na całym świe cie znaj dują się ser we ry lu strza ne z kodem źr ódłowym jądra Linuk sa. Trud no so bie obec nie wy obra zić Linuk sa bez standar do wej obsługi sie ci.

Utrzymywanie systemu

W ni niej szej książce będzie my mówić głównie o in sta la cji i kon fi gu ra cji. Jed na kże ad mi ni stra cja jest czy mś wię cej – po skon fi gu ro wa niu usługi mu sisz ta kże pil no wać, by działała. Wię k szość usług nie wy ma ga zbyt wiel kiej uwa gi, ale przy niektórych, ta kich jak pocz ta i gru py dys ku syj ne, mu sisz wy ko ny wać ru ty no we czyn no ści, by twój sys tem był spraw ny. Za da nia te omówimy w ko lej nych roz działach.

Ab so lut nym mi ni mum nie zbęd nym do po praw ne go funk cjo no wa nia sys te mu jest re gu lar ne spraw dza nie plików log sys te mu oraz plików log ka ż dej apli ka cji w ce lu wy kry cia błęd ów czy nie ty po wych zda rzeń. Zwy kle pi sze się w tym ce lu skryp ty ad mi ni stra cyj ne i co ja kiś czas uru cha mia się je z usługi *cron*. · ródłowe dys try bu cje nie któ ry ch wię k szych apli ka cji, ta kich jak *inn* czy C News, za wie rają ta kie skryp ty. Mu sisz tyl ko do pa so wać je do swo ich po trzeb.

Wy nik wszel kich za dań wy ko ny wa nych przez usługę *cron* po wi nien być wy syłany pocztą elek tro niczną na kon to ad mi ni stra cyj ne. Do my śl nie wie le apli ka cji wy syła ra por ty o błęd ach, sta ty sty ki wy ko rzy sta nia czy stresz cze nia plików log na kon to **root**. Ma to sens tyl ko wte dy, je żeli czę sto lo gu jesz się ja ko **root**. Du żo le piej jest przekazywać pocztę użytkownika **root** na własne konto, ustawiając alias pocztowy według opi su w roz dzia le 19, *Exim*, lub roz dzia le 18, *Sendmail*.

Choćbyśskonfigurował swój ośro dek z naj wię kszą dbałością, zgod nie z pra wa mi Mur ph y'ego i tak *wystąpi* ja kiś pro blem. Dla te go utrzy my wa nie sys te mu ozna cza także przyj mo wa nie skarg. Zwy kle lu dzie spo dzie wają się, że z ad mi ni stra to rem systemu mo żna skon tak to wać się pocztą elek tro niczną pod ad re sem *root*, ale ist nieją także in ne ad re sy, kt óre są po wszech nie uży wa ne do kon tak tu z oso ba mi od po wiedzia l ny mi za kon kret ny aspekt utrzy ma nia ośrod ka. Na przykład skar gi na te mat błęd nej kon fi gu ra cji pocz ty zwy kle będą wy syłane na ad res *postmaster*, a pro ble my z grupami dyskusyjnymi mogą być raportowane na ad res *newsmaster* lub *usenet*. Pocz ta na ad res *hostmaster* po win na być prze kie ro wa na do oso by od po wie dzial nej za pod sta wo we usługi sie cio we ho sta i usługi DNS, je żeli na two jej ma szy nie działa ser wer nazw.

Bezpieczeństwosystemu

Kolejnym, bardzo istotnym aspektem administracji systemu w środowisku sieciowym jest zabezpieczenie go i jego użytkowników przed intruzami. Niedbale zarządza ne sys te my sta no wią łatwy cel dla złośli wych osób. Ata ki za czy nają się od zga dy wa nia ha seł, a ko ńczą na wy syłaniu fałszy wych pak ietów Et her net, na to miast znisz cze nia za czy nają się od fałszy wych poczt elek tro nicz nych, a mogą sko ńczyć się utratą danych lub po gwałce niem pry wat no ści two ich uży tkowników. O pew nych

konkretnych problemach powiemy przy omawianiu kontekstu, w którym mogą one wystąpić, i pokażemy sposoby obrony.

Ten podrozdział omawia kilka przykładów i podstawowych technik związanych z bezpieczeństwem systemu. Oczywiście nie przedstawia wszystkich zagadnień bezpieczeństwa, jakie możesz napotkać. Chcemy jedynie zasygnalizować problemy, które mogą wystąpić. Dlatego przeczytanie dobrej książki na temat bezpieczeństwa jest absolutnie niezbędne, szczególnie w przypadku systemu sieciowego.

Podstawą bezpieczeństwa systemu jest dobra administracja. Oznacza to sprawdzanie własności wszystkich istotnych plików i katalogów oraz prawa dostępu do nich, a także monitorowanie wykorzystania uprzywilejowanych kont. Na przykład program COPS przeszukuje twój system plików i podstawowe pliki konfiguracyjne pod kątem nietypowych praw dostępu lub innych anomalii. Mądrze jest także używać takiego systemu haseł, który wymaga od użytkowników stosowania się do pewnych reguł, przez co hasła jest trudno odgadnąć. Na przykład pakiet haseł shadow wymaga, by hasło miało co najmniej 5 znaków i zawierało liczby oraz znaki nienumeryczne.

Gdy udostępniasz jakąś usługę w sieci, pamiętaj, żeby dać jej „jak najmniej przywilejów". Pozwalaj na robienie tylko tych rzeczy, które są wymagane, by działała tak, jak została zaprojektowana. Na przykład powinieneś nadać programom prawo setuid **roota** lub innego uprzywilejowanego konta, tylko wtedy gdy jest to niezbędne. Także, jeżeli chcesz używać usługi tylko w bardzo ograniczonym zakresie, nie wahaj się jej skonfigurować odpowiednio do twoich szczególnych zastosowań. Na przykład gdybyś chciał pozwolić, aby stacje bezdyskowe uruchamiały się z twojej maszyny, musisz udostępnić *uproszczony protokół przesyłania plików* (*Trivial File Transfer Protocol* – TFTP), tak by mogły skopiować podstawowe pliki konfiguracyjne z katalogu */boot* twojej maszyny. Jednak w przypadku nieograniczonego użycia TFTP pozwala użytkownikom z całego świata kopiować te pliki z twojego systemu, do których wszyscy mają prawo odczytu. Jeżeli sobie tego nie życzysz, ogranicz usługę TFTP jedynie do katalogu */boot**.

Możesz również ograniczyć usługi przyznawane użytkownikom określonych hostów, powiedzmy z twojej sieci lokalnej. W rozdziale 12 przedstawiamy demon *tcpd*, który wykonuje to zadanie dla wielu aplikacji sieciowych. Bardziej wyrafinowane metody ograniczania dostępu do poszczególnych hostów lub usług omówimy w rozdziale 9.

Kolejną ważną rzeczą jest unikanie „niebezpiecznego" oprogramowania. W pewnym sensie każde oprogramowanie może być niebezpieczne, ponieważ może zawierać błędy, które sprytni ludzie mogą wykorzystać, by uzyskać dostęp do twojego systemu. Takie rzeczy się zdarzają i nie da się przed tym zabezpieczyć. Problem ten dotyczy zarówno oprogramowania darmowego, jak i produktów komercyjnych**. Jednak programy wymagające specjalnych przywilejów są z natury bardziej nara-

* Do tego tematu powrócimy w rozdziale 12, *Ważne funkcje sieciowe*.

** Zdarzały się komercyjne wersje Uniksa (za które płaciło się mnóstwo pieniędzy), których skrypty powłoki miały tak ustawione prawo setuid **root**, że użytkownik mógłby bez trudu uzyskać przywileje **roota** za pomocą standardowej sztuczki.

żone na niebezpieczeństwo niż pozostałe, ponieważ wszelkie luki mogą prowadzić do poważnych konsekwencji*. Jeżeli instalujesz program z prawem setuid, który ma pracować z siecią, bądź dwa razy bardziej ostrożny i przeczytaj dokumentację, abyś przez przypadek nie stworzył dziury w bezpieczeństwie.

Uwagę powinieneś zwrócić także na programy, które pozwalają na logowanie lub wykonywanie poleceń z niepełnym uwierzytelnianiem. Polecenia, takie jak *rlogin*, *rsh* i *rexec*, są bardzo przydatne, ale od osoby uruchamiającej wymagają jedynie ograniczonego uwierzytelnienia, które opiera się na zaufaniu do nazwy wywołującego hosta, ustalonej na podstawie serwera nazw (będziemy mówili o tym później), którą łatwo można sfałszować. Obecnie standardową praktyką powinno być zupełne wyłączanie poleceń *r* i zastępowanie ich narzędziami z pakietu *ssh*. Narzędzia *ssh* wykorzystują bardziej niezawodne metody uwierzytelniania i oferują także inne usługi, takie jak szyfrowanie i kompresja.

Nigdy nie możesz wykluczyć możliwości, że twoje zabezpieczenia kiedyś zawiodą, bez względu na to, jak byłeś ostrożny. Dlatego powinieneś upewnić się, że dostatecznie wcześnie wykrywasz intruzów. Sprawdzanie logów systemu jest dobrym punktem początkowym, ale intruz jest prawdopodobnie wystarczająco mądry, by przewidzieć, że tak postąpisz, i usunie wszelkie oczywiste ślady pozostawione przez siebie. Jednak istnieją narzędzia takie jak *tripwire*, napisane przez Gene Kima i Gene Spafforda, które pozwalają sprawdzać istotne pliki systemu, by zobaczyć, czy ich zawartość lub prawa dostępu nie zostały zmienione; *tripwire* liczy różne sumy kontrolne tych plików i umieszcza je w bazie danych. W czasie kolejnych przebiegów sumy są liczone ponownie i porównywane z wcześniej zapisanymi, by w ten sposób wykryć modyfikacje.

* W 1988 roku z powodu błędu RTM nieomal doszło do zablokowania Internetu, częściowo przez wykorzystanie dziury w pewnych programach, między innymi w *sendmailu*. Dziura ta istniała przez dość długi czas, zanim została załatana.



2

Wybrane problemy sieci TCP/IP

W tym rozdziale powiemy, jakie decyzje konfiguracyjne musisz podjąć, jeśli chcesz podłączyć swojego Linuksa do sieci TCP/IP. Zajmiemy się adresami IP, nazwami hostów i rutingiem. Rozdział ten daje ci podstawową wiedzę, niezbędną do zrozumienia, czego wymaga twoja konfiguracja, natomiast w następnych rozdziałach poznasz narzędzia, których będziesz używał.

Aby dowiedzieć się więcej o TCP/IP i jego budowie, zajrzyj do trzytomowej książki *Internetworking with TCP/IP* Douglasa R. Comera (Prentice Hall). Bardziej szczegółowym przewodnikiem po zarządzaniu siecią TCP/IP jest książka *TCP/IP Network Administration* Craiga Hunta (O'Reilly).

Interfejsy sieciowe

Aby ukryć różnorodność sprzętu obecnego w środowisku sieciowym, TCP/IP odwołuje się do *interfejsu*, przez który następuje dostęp do sprzętu. Interfejs oferuje zestaw operacji identyczny dla wszystkich rodzajów urządzeń; za jego pomocą obsługuje się wysyłanie i odbieranie pakietów.

Każde sieciowe urządzenie peryferyjne musi mieć w jądrze odpowiedni interfejs. Na przykład interfejsy Ethernet w Linuksie noszą nazwy *eth0* i *eth1*, interfejsy PPP (omówione w rozdziale 8, *Protokół punkt-punkt*) są nazywane *ppp0* i *ppp1*, a interfejsy FDDI – *fddi0* i *fddi1*. Nazwy interfejsów są używane tylko w poleceniu konfiguracyjnym, kiedy chcesz się odwołać do konkretnego urządzenia fizycznego. Poza tym nie są stosowane.

Zanim interfejsu będzie można użyć w sieci TCP/IP, należy mu przypisać adres IP, który identyfikuje go w procesie komunikacji z resztą świata. Adres ten jest różny od wspomnianej poprzednio nazwy interfejsu. Jeżeli porównasz interfejs do drzwi, to adres jest przyczepioną na nich tabliczką z nazwiskiem.

Można ustawiać także inne parametry urządzenia, takie jak maksymalny rozmiar datagramów (*Maximum Transfer Unit* – MTU), które mogą być przetworzone przez konkretne urządzenie. Inne atrybuty omówimy później. Na szczęście większość atrybutów ma sensowne wartości domyślne.

Adresy IP

Jak wspomnieliśmy w rozdziale 1, *Wprowadzenie do sieci*, protokół sieciowy IP rozumie adresy w postaci liczb 32-bitowych. Każda maszyna musi mieć przypisany numer, który jest niepowtarzalny w środowisku sieciowym*. Jeżeli jesteś podłączony do sieci lokalnej, która nie wymienia ruchu TCP/IP z innymi sieciami, możesz przypisać adresy zgodnie z własnym widzimisię. Istnieją pewne zakresy adresów IP, które zostały zarezerwowane dla takich sieci prywatnych. Pokazano je w tabeli 2-1. Jednak ośrodkom podłączonym do Internetu adresy są nadawane przez administrację centralną: NIC (*Network Information Center*)**.

Adresy IP, aby były łatwo czytelne, są podzielone na cztery 8-bitowe liczby, zwane oktetami. Na przykład **quark.physics.groucho.edu** ma adres IP **0x954C0C04**, zapisywany jako **149.76.12.4**. Format ten często nazywany jest *kropkową notacją czwórkową*.

Innym powodem zastosowania takiego zapisu jest to, że adresy IP są dzielone na *numer sieci* zawarty w pierwszej części adresu i na *numer hosta* zawarty w pozostałej jego części. Gdy prosisz NIC o adresy IP, nie dostajesz adresu dla każdego hosta, który planujesz podłączyć. Otrzymujesz numer sieci i pozwolenie na utworzenie w przyznanym zakresie prawidłowych adresów IP dla hostów w twojej sieci, zgodnie z potrzebami.

Rozmiar części sieciowej adresu zależy od wielkości sieci. Aby uwzględnić różne potrzeby, zdefiniowano kilka klas sieci dzielących adresy IP w różnych miejscach. Klasy sieci są następujące:

Klasa A
: Klasa A obejmuje sieci od **1.0.0.0** do **127.0.0.0**. Numer sieci jest zapisany w pierwszym oktecie. Klasa ta udostępnia 24-bitowy adres hosta, co pozwala na podłączenie do jednej sieci, z grubsza rzecz biorąc, 1,6 miliona hostów w każdej sieci.

Klasa B
: Klasa B obejmuje sieci od **128.0.0.0** do **191.255.0.0**. Numer sieci jest zapisany w dwóch pierwszych oktetach. Klasa ta pozwala na stworzenie 16 320 sieci o 65 024 hostach w każdej z nich.

* Najczęściej używa się 4. wersji protokołu IP. Wiele wysiłku włożono w opracowanie jej rozszerzenia oznaczonego jako wersja 6. IPv6 używa innego schematu adresowania i dłuższych adresów. Linux posiada implementację IPv6, ale nie jest ona jeszcze na tyle dopracowana, by dokumentować ją w tej książce. Obsługa IPv6 w jądrze Linuksa jest dobra, ale należy zmodyfikować wiele aplikacji sieciowych, by także obsługiwały ten standard. Cierpliwości.

** Zwykle adresy IP nadaje usługodawca, u którego kupuje się połączenie IP. Jednak można się także zgłosić po adresy IP bezpośrednio do NIC, wysyłając e-mail pod adresem *hostmaster@internic.net* lub używając formularza znajdującego się pod adresem *http://www.internic.net/*)

Klasa C

Klasa C obejmuje sieci od **192.0.0.0** do **223.255.255.0**, gdzie numer sieci jest zapisany w trzech pierwszych oktetach. Klasa ta pozwala na zarejstrowanie prawie 2 milionów sieci po 254 hosty w każdej.

Klasy D, E i F

Adresy należące do zakresu **224.0.0.0** do **254.0.0.0** są albo eksperymentalne, albo zarezerwowane do zastosowań specjalnych i nie określają żadnej sieci. Transmisja grupowa IP (ang. *IP multicasting*) – usługa pozwalająca na przesyłanie danych do wielu miejsc w Internecie jednocześnie – wymaga przypisania adresów właśnie z tego zakresu.

Jeśli wrócimy do przykładu z rozdziału 1, stwierdzimy, że **149.76.12.4** (adres **quarka**) oznacza host o numerze **12.4** w sieci klasy B o numerze **149.76.0.0**.

Być może zauważyłeś przy opisie klas adresów, że nie wszystkie możliwe wartości były dozwolone dla każdego oktetu w części opisującej hosta. Dzieje się tak dlatego, że oktety **0** i **255** są zarezerwowane do specjalnych celów. Adres, w którym wszystkie bity w części hosta mają wartość 0, jest adresem sieci, a adres, w którym wszystkie bity w części hosta mają wartość 1, nazywa się *adresem rozgłoszeniowym* (ang. *broadcast address*). Odnosi się on do wszystkich hostów w zadanej sieci jednocześnie. Tak więc **149.76.255.255** nie jest poprawnym adresem hosta, ale odnosi się do wszystkich hostów w sieci **149.76.0.0**.

Kilka adresów sieci jest zarezerwowane do szczególnych celów. Dwa takie adresy to: **0.0.0.0** i **127.0.0.0**. Pierwszy nazywamy *domyślnym rutingiem* (ang. *default route*), a drugi *adresem pętli zwrotnej* (ang. *loopback address*). Domyślny ruting jest związany ze sposobem kierowania datagramów IP.

Sieć **127.0.0.0** jest zarezerwowana dla ruchu IP lokalnego względem twojego hosta. Zwykle adres **127.0.0.1** zostaje przypisany specjalnemu interfejsowi twojego hosta – *interfejsowi pętli zwrotnej*, który działa jak obwód zamknięty. Dowolny pakiet IP skierowany na ten interfejs z TCP lub UDP zostanie mu zwrócony tak, jak by właśnie nadszedł z jakiejś sieci. Dzięki temu można testować oprogramowanie sieciowe bez wykorzystywania „rzeczywistej” sieci. Sieć pętli zwrotnej pozwala także na użycie oprogramowania sieciowego na pojedynczym hoście. Choć nie wygląda to na zbyt przydatne, to jednak jest. Na przykład wiele ośrodków UUCP nie posiada w ogóle podłączenia IP, ale wciąż może w nich działać system grup dyskusyjnych INN. Aby prawidłowo pracować w Linuksie, INN wymaga interfejsu pętli zwrotnej.

W każdej klasie sieci pewne zakresy adresów zostały odłożone na bok i określone jako „zarezerwowane” lub „prywatne” zakresy adresów. Adresy te są przeznaczone do użytku w sieciach prywatnych i nie są rutowane do Internetu. Zwykle korzystają z nich organizacje tworzące własny intranet, ale także małe sieci. Jak już mówiliśmy, zarezerwowane adresy sieci podaje tabela 2-1.

Tabela 2-1. Zakresy adresów IP zarezerwowane do użytku prywatnego

Klasa	*Sieci*
A	10.0.0.0 do 10.255.255.255
B	172.16.0.0 do 172.31.0.0
C	192.168.0.0 do 192.168.255.0

Rozwiązywanie adresów

Teraz, gdy wiesz już, jak tworzy się adresy IP, możesz zastanawiać się, w jaki sposób są one używane do adresowania hostów w sieci Ethernet lub Token Ring. Przecież protokoły te mają własne adresy identyfikujące hosty, które nie mają nic wspólnego z adresem IP. Prawda? Tak, masz rację.

Potrzebny jest mechanizm, który odwzorowuje adresy IP na adresy sieci niższej warstwy. Tym mechanizmem jest *protokół rozwiązywania adresów* (*Address Resolution Protocol* – ARP). W praktyce ARP można stosować nie tylko w Ethernecie czy Token Ringu, ale również w innych typach sieci, między innymi takich, w których pracuje protokół AX.25. Metoda działania ARP jest taka sama jak ta, którą posługuje się większość ludzi, kiedy musi znaleźć Pana X wśród 150 gości: osoba szukająca krzyczy na tyle głośno, by każdy mógł ją usłyszeć, i oczekuje, że jeżeli Pan X jest wśród zgromadzonych, to się odezwie. Gdy X odpowie, wiemy, która to osoba.

Gdy ARP chce znaleźć adres ethernetowy odpowiadający określonemu adresowi IP, wykorzystuje funkcję Ethernetu zwaną *rozgłaszaniem* (ang. *broadcasting*). Rozgłaszanie polega na tym, że datagram jest adresowany do wszystkich stacji w sieci jednocześnie. Datagram rozgłoszeniowy wysłany przez ARP zawiera zapytanie o adres IP. Każdy host, który go odbierze, porównuje to zapytanie ze swoim własnym adresem IP. Jeżeli znajdzie się host, którego adres IP odpowiada poszukiwanemu, to zwraca on odpowiedź ARP do pytającego hosta. Pytający host może teraz – na podstawie odpowiedzi – odczytać adres ethernetowy nadawcy.

Możesz się zastanawiać, jak host może uzyskać adres innego hosta, który znajduje się na przykład w zupełnie innej sieci na drugim końcu świata. Odpowiedź na to pytanie wymaga wyjaśnienia mechanizmu *rutingu*, czyli znalezienia fizycznej lokalizacji hosta w sieci. To zagadnienie omówimy dokładniej w następnym podrozdziale.

Powiedzmy nieco więcej o protokole ARP. Gdy host znajdzie adres ethernetowy, zapisuje go w pamięci podręcznej ARP, aby nie pytać o niego, gdy będzie chciał ponownie wysłać datagram do tego samego hosta. Jednak niemądre byłoby trzymanie tej informacji bez końca. Karta Ethernet zdalnego hosta może zostać wymieniona z powodów technicznych, a więc wpis ARP byłby błędny. Dlatego wpisy w pamięci podręcznej ARP są po pewnym czasie usuwane, by wymusić kolejne zapytanie o adres IP.

Czasem trzeba również znaleźć adres IP odpowiadający danemu adresowi ethernetowemu. Dzieje się tak, gdy maszyna bezdyskowa chce uruchomić się z serwera w sieci. Sytuacja ta jest powszechnie spotykana w sieciach LAN. Jednak klient bezdyskowy nie ma o sobie informacji – za wyjątkiem swojego adresu Ethernet. Tak więc

rozgłasza komunikat, w którym prosi serwer uruchomieniowy (ang. *boot server*) o adres IP. Tę sytuację obsługuje protokół o nazwie *odwrotny protokół rozwiązywania adresów* (*Reverse Address Resolution Protocol* – RARP). Wraz z protokołem BOOTP pozwala na uruchamianie bezdyskowych klientów z sieci.

Ruting IP

Teraz wyjaśnijmy, jak na podstawie adresu IP odszukać host, do którego są adresowane datagramy. Różne części adresu są obsługiwane w różny sposób. Twoim zadaniem jest skonfigurowanie plików tak, aby mówiły, jak ma być traktowana każda z poszczególnych części adresu IP.

Sieci IP

Gdy piszesz do kogoś list, zwykle umieszczasz na kopercie pełny adres, czyli także państwo, region administracyjny (np. stan, województwo), na zwę pocz ty wraz z kodem. Po włożeniu listu do skrzynki pocztowej, poczta dostarczy go do miejsca przeznaczenia: zostanie wysłany do podanego na kopercie kraju, gdzie służby krajowe skierują go do odpowiedniego regionu. Zaleta takiego hierarchicznego schematu jest oczywista: gdy wysyłasz list do innego miasta lub kraju, miejscowa poczta wie z grubsza, w jakim kierunku ma go przekazać, ale nie martwi się, którędy list będzie szedł, gdy już dotrze do kraju przeznaczenia.

Sieci IP mają podobną strukturę. Cały Internet składa się z szeregu sieci, zwanych *systemami niezależnymi*. Każdy system realizuje wewnętrznie ruting pomiędzy swoimi hostami, tak więc zadanie dostarczenia datagramu redukuje się do znalezienia ścieżki do sieci zawierającej host adresata. Wystarczy przekazać datagram do *jakiegokolwiek* hosta w sieci adresata, a dalsza wędrówka odbywa się już wyłącznie w obrębie tej sieci.

Podsieci

Zasada podziału jest widoczna w wyodrębnieniu w adresach IP części hosta i części sieciowej, jak już wyjaśnialiśmy wcześniej. Domyślnie sieć przeznaczenia jest uzyskiwana z części sieciowej adresu IP. Tak więc hosty o identycznych numerach *sieci* IP powinny znajdować się w tej samej sieci*.

Zastosowanie podobnego schematu ma także sens *wewnątrz* sieci, ponieważ może się ona składać z setek mniejszych sieci, w których najmniejszymi jednostkami są fizyczne sieci np. Ethernet. Dlatego IP pozwala na dalszy podział sieci IP na kilka *podsieci*.

Podsieć odpowiada za dostarczanie datagramów do pewnego zakresu adresów IP. Jest to rozszerzenie pojęcia podziału pól bitowych, tak jak w klasach A, B i C. Jednak część sieciowa jest teraz rozszerzana tak, by zawierała niektóre bity z części hosta. Liczba bitów, interpretowana jako numer podsieci, jest określona przez tak zwaną

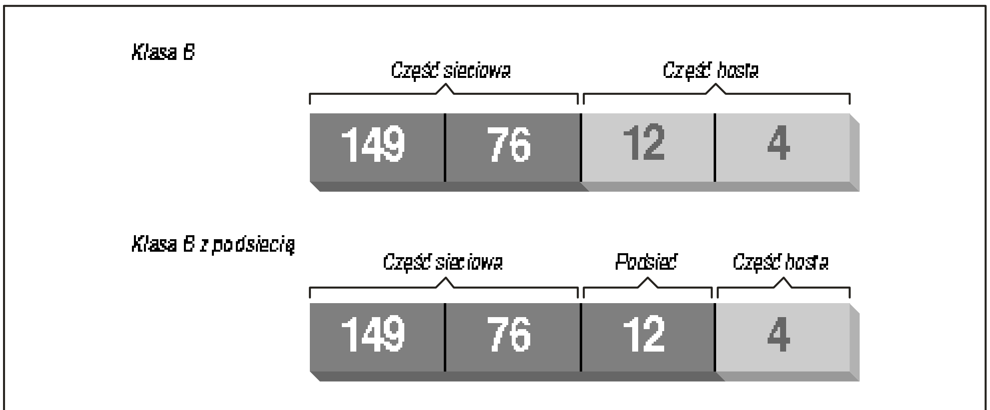
* Systemy niezależne są nieco bardziej ogólne. Mogą składać się z więcej niż jednej sieci IP.

maskę podsieci lub *maskę sieci*. Jest to również liczba 32-bitowa, określająca maskę bitową dla części sieciowej adresu IP.

Sieć campusowa przykładowego uniwersytetu Groucho Marx to właśnie taka sieć. Ma sieć klasy B o numerze **149.76.0.0** i dlatego jej maska to **255.255.0.0**.

Wewnętrznie sieć campusowa składa się z kilku mniejszych sieci, takich jak sieci lokalne różnych wydziałów. Tak więc zakres adresów IP jest podzielony na 254 podsieci od **149.76.1.0** do **149.76.254.0**. Na przykład wydział fizyki teoretycznej ma przypisany adres **149.76.12.0**. Szkielet campusu jest siecią samą w sobie i ma numer **149.76.1.0**. Podsieci te korzystają ze wspólnej części sieciowej adresu IP, a rozróżniane są na podstawie 3. oktetu. Dlatego maska podsieci w tym przypadku ma postać **255.255.255.0**.

Rysunek 2-1 pokazuje, jak adres **quarka**: **149.76.12.4** jest interpretowany, gdy ma być zwykłym adresem w sieci klasy B i wtedy, gdy ma uwzględniać podsieci.



Rysunek 2-1. Podział sieci klasy B na podsieci

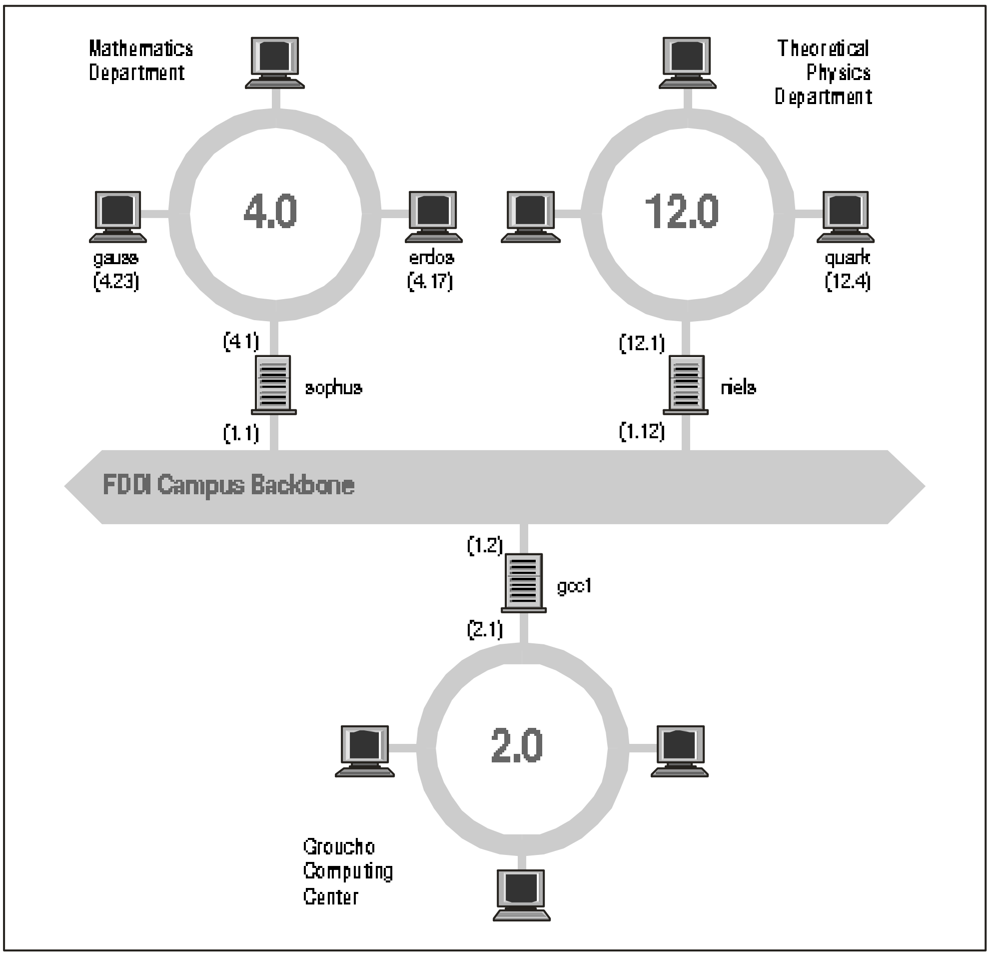
Warto zauważyć, że *dzielenie na podsieci* (technika tworzenia podsieci) jest jedynie *wewnętrznym podziałem* sieci. Podsieci są generowane przez właściciela sieci (lub administratorów). Często podsieci są tworzone, aby odzwierciedlać istniejące granice fizyczne (pomiędzy dwoma sieciami Ethernet), administracyjne (pomiędzy dwoma wydziałami) lub geograficzne (pomiędzy dwoma lokalizacjami), natomiast władza nad każdą z podsieci jest oddawana innej osobie. Jednak struktura ta dotyczy tylko wewnętrznego zachowania sieci i jest zupełnie niewidoczna dla świata zewnętrznego.

Gatewaye

Podział na podsieci jest korzystny nie tylko ze względów organizacyjnych. Często jest naturalną konsekwencją ograniczeń sprzętowych. Punkt widzenia hosta na daną sieć fizyczną, np. Ethernet, jest bardzo ograniczony: może on komunikować się tylko z hostem w sieci, do której jest podłączony. Dostęp do wszystkich innych hostów jest możliwy tylko przez przeznaczone do tego urządzenia nazywane *gateway-*

ami. Gateway to host, który jest podłączony do dwóch lub więcej sieci fizycznych jednocześnie i skonfigurowany tak, by przekazywać pakiety między tymi sieciami.

Rysunek 2-2 pokazuje fragment topologii sieci uniwersytetu Groucho Marx (GMU). Hosty należące jednocześnie do dwóch podsieci są opatrzone oboma adresami.



Rysunek 2-2. Fragment schematu sieci uniwersytetu Groucho Marx

Różne sieci fizyczne muszą być różnymi sieciami IP, aby protokół IP był w stanie rozpoznać, że host jest w sieci lokalnej. Na przykład numer sieci **149.76.4.0** jest zarezerwowany dla hostów w sieci LAN wydziału matematyki. Przy przesyłaniu datagramów do **quarka**, oprogramowanie sieciowe na **erdosie** natychmiast rozpozna je na podstawie adresu IP **149.76.12.4**, że host docelowy jest w innej sieci fizycznej, a co za tym idzie można się do niego dostać jedynie przez gateway (domyślnie **sophus**).

Sam **sophus** jest podłączony do dwóch różnych podsieci: wydziału matematyki i sieci szkieletowej campusu. Dostęp do każdej z nich ma przez różne interfejsy, odpowiednio *eth0* i *fddi0*. Jaki w takim razie powinniśmy przypisać mu adres IP? Powinien mieć adres z podsieci **149.76.1.0** czy może raczej z **149.76.4.0**?

Odpowiedź brzmi: oba. Gateway **sophus** ma przypisany adres **149.76.1.1** do użytku w sieci **149.76.1.0** i adres **149.76.4.1** do użytku w sieci **149.76.4.0**. Gateway musi mieć odrębny adres IP w każdej sieci, do której należy. Adresy te, wraz z odpowiadającymi im maskami sieci, są związane z interfejsem, przez który następuje dostęp do sieci. Tak więc odwzorowanie interfejsu na adres w przypadku **sophusa** wygląda następująco:

Interfejs	*Adres*	*Maska sieci*
eth0	149.76.4.1	255.255.255.0
fddi0	149.76.1.1	255.255.255.0
lo	127.0.0.1	255.0.0.0

Ostatnia pozycja to interfejs pętli zwrotnej *lo*, o którym pisaliśmy wcześniej.

Zwykle możesz zignorować subtelne różnice pomiędzy wiązaniem adresu z hostem lub jego interfejsem. Jeśli host jest tylko w jednej sieci, tak jak **erdos**, będziesz się odwoływał do hosta o takim a takim adresie IP, choć dokładnie rzecz biorąc, to interfejs Ethernet ma przypisany adres IP. Różnica jest naprawdę istotna tylko w przypadku gatewaya.

Tablica rutingu

Teraz skupimy się na tym, jak IP wybiera gateway, który ma zostać wykorzystany do dostarczenia datagramu do odległej sieci.

Widzieliśmy, że kiedy **erdos** otrzyma datagram przeznaczony dla **quarka**, sprawdza adres docelowy i stwierdza, że nie leży on w sieci lokalnej. Dlatego **erdos** wysyła datagram do domyślnego gatewaya **sophus**, przed którym stoi teraz to samo zadanie. **sophus** stwierdza, że nie ma takiego hosta w sieciach, do których jest bezpośrednio podłączony. Musi więc znaleźć inny gateway, do którego będzie mógł przekazać datagram. Poprawnym wyborem będzie **niels**, gateway wydziału fizyki. **sophus** potrzebuje zatem informacji wiążących docelową sieć z odpowiednim gatewayem.

IP wykorzystuje do tego celu tablicę, która łączy sieci z gatewayami, przez które można do nich dotrzeć. Musi w niej istnieć także wpis uniwersalny (*ruting domyślny*) – jest to gateway związany z siecią **0.0.0.0**. Wszystkie adresy docelowe pasują do tej trasy, ponieważ żaden z 32 bitów nie musi odpowiadać temu wpisowi i dlatego pakiety do nieznanej sieci są wysyłane przez trasę domyślną. Dla gatewaya **sophusa**, tablica mogłaby wyglądać tak:

Sieć	*Maska sieci*	*Gateway*	*Interfejs*
149.76.1.0	255.255.255.0	-	*fddi0*
149.76.2.0	255.255.255.0	149.76.1.2	*fddi0*
149.76.3.0	255.255.255.0	149.76.1.3	*fddi0*
149.76.4.0	255.255.255.0	-	*eth0*
149.76.5.0	255.255.255.0	149.76.1.5	*fddi0*
...	...	...	...
0.0.0.0	0.0.0.0	149.76.1.2	*fddi0*

Jeżeli masz skorzystać z trasy do tej sieci, do której **sophus** jest bezpośrednio podłączony, nie potrzebujesz gatewaya. Kolumna z wpisem gatewaya w takim przypadku zawiera kreskę.

Proces identyfikacji, czy dany adres docelowy pasuje do trasy, jest operacją matematyczną. Jest dość prosty, ale wymaga znajomości logiki i arytmetyki binarnej: żądana trasa pasuje do trasy docelowej, jeżeli adres sieci po wykonaniu logicznej operacji AND z maską sieci jest dokładnie taki sam, jak adres docelowy po wykonaniu operacji logicznej AND z maską sieci.

Wyjaśnienie: trasa jest prawidłowa, jeżeli liczba bitów adresu sieci określona przez maskę sieci (począwszy od pierwszego bitu leżącego od lewej strony, czyli najstarszego bitu pierwszego bajtu adresu) jest taka sama jak liczba bitów w adresie docelowym.

Gdy implementacja IP poszukuje najlepszej trasy do miejsca docelowego, może znaleźć wiele pasujących wpisów z trasami. Na przykład wiemy, że domyślny ruting pasuje do każdego adresu docelowego, ale datagramy kierowane do sieci podłączonych lokalnie będą pasowały także do własnych tras. Skąd IP wie, której trasy użyć? To właśnie tutaj maska sieci ma decydujące znaczenie. Choć obie trasy pasują do adresu docelowego, jedna z nich ma większą maskę sieci niż druga. Wspomnieliśmy wcześniej, że maska sieci była używana do podziału naszej przestrzeni adresowej na mniejsze sieci. Im większa jest maska, tym lepiej jest dopasowywany adres docelowy. Wyznaczając trasę dla datagramu powinniśmy zawsze wybierać trasę o największej masce sieci. Domyślna trasa ma maskę sieci o wielkości 0 bitów, a w powyżej pokazanej konfiguracji, lokalnie podłączone sieci mają maski sieci o długości 24 bitów. Jeżeli datagram odpowiada lokalnie podłączonej sieci, będzie rutowany w pierwszej kolejności do odpowiedniego urządzenia, a nie na adres domyślny, gdyż lokalne trasy są dopasowane większą liczbą bitów. Tylko te datagramy, które nie pasują do żadnej trasy, będą przesyłane przez trasę domyślną.

Tablice rutingu możesz tworzyć na różne sposoby. Dla małych sieci lokalnych zwykle najlepiej przygotować ją ręcznie i udostępnić protokołowi IP za pomocą polecenia *route* w czasie uruchamiania maszyny (zobacz rozdział 5, *Konfigurowanie sieci TCP/IP*). Dla większych sieci tablice są budowane i uzupełniane w czasie pracy sieci przez *demony rutingu*; te programy pracują na centralnych hostach sieci i wymieniają informacje o rutingu, by obliczyć „optymalne" trasy pomiędzy podłączonymi sieciami.

Rozmiar sieci decyduje też o wyborze protokołów rutingu. W przypadku rutingu w systemach niezależnych (tak jak w campusie Groucho Marx), używane są *wewnętrzne protokoły rutingu*. Najbardziej znanym z nich jest RIP (*Routing Information Protocol*), zaimplementowany w demonie *routed* BSD. W przypadku rutingu pomiędzy systemami autonomicznymi stosowane są *zewnętrzne protokoły rutingu*, takie jak EGP (*External Gateway Protocol*) lub BGP (*Border Gateway Protocol*). Protokoły te, wraz z RIP-em, zostały zaimplementowane w demonie *gated* napisanym na Uniwersytecie Cornella.

Wartości metryki

Można skorzystać z rutingu dynamicznego, jeżeli trzeba znaleźć najlepszą trasę do hosta docelowego lub sieci na podstawie liczby *hopów*. Hopy oznaczają liczbę gatewayów, przez które datagram musi przejść, zanim dotrze do hosta lub sieci. Im krótsza jest trasa, tym lepiej radzi sobie z nią RIP. Bardzo długie trasy (ponad 16 hopów) są traktowane jako bezużyteczne i są usuwane.

RIP obsługuje informacje o rutingu wewnątrz twojej sieci lokalnej, ale na wszystkich hostach musisz uruchomić demona *gated*. W czasie startu komputera *gated* sprawdza wszystkie aktywne interfejsy sieciowe. Jeżeli jest aktywny więcej niż jeden interfejs (nie licząc interfejsu pętli zwrotnej), demon zakłada, że host przekazuje pakiety pomiędzy kilkoma sieciami i czynnie wymienia oraz rozgłasza informacje o rutingu. W przeciwnym razie jedynie pasywnie odbiera uaktualnienia RIP i odświeża lokalną tablicę rutingu.

Przy rozgłaszaniu informacji z lokalnej tablicy rutingu, *gated* liczy długość trasy na podstawie tak zwanej *wartości metryki* (ang. *metric value*) związanej z wpisem w tablicy. Ta wartość jest ustawiana przez administratora podczas konfigurowania rutingu i powinna odpowiadać rzeczywistemu kosztowi trasy*. Dla tego metryka trasy do podsieci, do której host jest podłączony bezpośrednio, zawsze powinna wynosić zero, natomiast trasa prowadząca przez dwa gatewaye powinna mieć metrykę o wartości dwa. Nie musisz przejmować się metryką, jeżeli nie używasz protokołu *RIP*-a ani *gated*.

Internetowy protokół komunikatów kontrolnych (ICMP)

IP ma protokół towarzyszący, o którym jeszcze nie mówiliśmy. Jest nim ICMP (*Internet Control Message Protocol*) używany przez kod sieciowy jądra do przesyłania komunikatów o błędach do innych hostów. Na przykład załóżmy, że jesteś znów na **erdosie** i chcesz zrealizować połączenie *telnet* z portem 12345 na **quarku**, ale na tym porcie nie ma procesu nasłuchującego. Gdy pierwszy pakiet TCP zaadresowany na ten port nadejdzie do **quarka**, warstwa sieciowa rozpozna, że coś przyszło i natychmiast zwróci do **erdosa** komunikat ICMP o treści „Port Unreachable" (port nieosiągalny).

* Koszt trasy to, w prostych sieciach, liczba hopów wymaganych do dotarcia do celu. W bardziej skomplikowanych sieciach poprawne obliczenie kosztu trasy może być trudne.

Protokół ICMP udostępnia różne komunikaty, głównie z informacjami o błędach. Jednak istnieje jeden ciekawy komunikat, tak zwany komunikat przekierowania (ang. *redirect message*). Jest on generowany przez moduł rutingu, gdy wykryje on, że inny host używa naszego hosta jako gatewaya, mimo że istnieje krótsza trasa. Na przykład po uruchomieniu systemu tablica rutingu na **sophusie** może być niepełna. Może zawierać trasy do sieci wydziału matematyki, do szkieletu FDDI i domyślną trasę do gatewaya centrum obliczeniowego Groucho (**gcc1**). Tak więc pakiety adresowane do **quarka** będą wysyłane do **gcc1**, a nie do **nielsa** – gatewaya wydziału fizyki. Po odebraniu takiego datagramu **gcc1** zauważy, że jest to nieoptymalna trasa i przekaże pakiet do **nielsa**, zwracając równocześnie do **sophusa** komunikat przekierowania ICMP z informacją o lepszej trasie.

Wydaje się, że w ten sposób można łatwo uniknąć ręcznej konfiguracji wszelkich tras poza podstawowymi. Trzeba jednak zdawać sobie sprawę, że poleganie na schematach rutingu dynamicznego, czy to będzie RIP, czy komunikat przekierowania ICMP, nie zawsze jest dobre. Przekierowanie ICMP i RIP dają ci niewielką możliwość (lub wręcz nie dają ci żadnej szansy) weryfikowania pokrywających się informacji o rutingu. Ta sytuacja może prowadzić do zakłócenia pracy całej twojej sieci lub jeszcze gorszych rzeczy. W rezultacie kod sieciowy Linuksa traktuje komunikaty przekierowania sieci tak, jakby to były przekierowania hosta. Minimalizuje to zniszczenia w przypadku ataku, które dotkną wówczas jeden host, a nie całą sieć. Z drugiej strony oznacza to, że w przypadku legalnej sytuacji generowany jest nieco większy ruch, gdyż każdy host wysyła komunikat przekierowania ICMP. Obecnie opieranie się na przekierowaniach ICMP nie jest dobrze widziane i uznaje się je raczej za złą praktykę.

Rozwiązywanie nazwy hosta

Jak wcześniej napisaliśmy, adresowanie w sieci TCP/IP, przynajmniej tam, gdzie korzysta się z IP w wersji 4, opiera się na liczbach 32-bitowych. Nie ukrywamy, że zapamiętywanie takich liczb nie jest łatwe. Dlatego hosty występują również pod „zwykłymi" nazwami, takimi jak **gauss** czy **strange**. Znalezienie adresu IP odpowiadającego nazwie to obowiązek aplikacji. Proces ten jest nazywany *rozwiązywaniem nazwy hosta*.

Gdy aplikacja chce znaleźć adres IP danego hosta, korzysta z funkcji bibliotecznej *gethostbyname(3)* i *gethostbyaddr(3)*. Tradycyjnie te i inne związane z nimi procedury były zgrupowane w oddzielnej bibliotece o nazwie *resolver library*. W Linuksie funkcje te są częścią standardowej biblioteki *libc*. Potocznie zestaw tych funkcji jest nazywany „resolverem". Konfigurację mechanizmu rozwiązywania nazw opisano szczegółowo w rozdziale 6, *Usługi nazewnicze i konfigurowanie resolvera*.

W przypadku małej sieci Ethernet czy nawet grupy takich sieci, nie jest trudno utrzymywać tablice odwzorowujące nazwy hostów na adresy. Informacja ta jest zwykle przechowywana w pliku o nazwie */etc/hosts*. Podczas dodawania lub usuwania hostów albo zmiany przypisania adresów, wystarczy uaktualnić plik *hosts* na wszystkich

hostach. Oczywiście staje się to uciążliwe przy sieciach, które składają się z więcej niż kilku maszyn.

Jednym z rozwiązań jest NIS (*Network Information System* – system informacji sieciowej) stworzony przez firmę Sun Microsystems, potocznie nazywany YP lub Yellow Pages. NIS przechowuje plik *hosts* (i inne informacje) w bazie danych na hoście głównym, z którego klienty mogą go w razie potrzeby odczytywać. Rozwiązanie takie jest odpowiednie jedynie dla średniej wielkości sieci typu LAN, ponieważ wymaga utrzymania centralnej bazy danych *hosts* i dystrybuowania jej do wszystkich serwerów. Instalacja i konfiguracja NIS-a została omówiona w rozdziale 13, *System informacji sieciowej*.

W Internecie informacje adresowe były pierwotnie przechowywane także w pliku bazy danych *HOSTS.TXT*. Plik ten był utrzymywany przez NIC (*Network Information Center* – centrum informacji sieciowej) i musiał być stamtąd pobierany i instalowany przez wszystkie ośrodki podłączone do Internetu. Gdy sieć się rozrosła, takie rozwiązanie stało się niewygodne. Poza uciążliwym w administracji regularnym instalowaniem pliku *HOSTS.TXT*, niebezpiecznie wzrosło obciążenie dystrybuujących go serwerów. Co więcej, wszystkie nazwy musiały być rejestrowane w NIC, aby mieć pewność, że żadna się nie powtarza.

Dlatego w 1994 roku przyjęto nowy schemat rozwiązywania nazw: *system nazw domen* (*Domain Name System* – DNS) autorstwa Paula Mockapetrisa. System nazw domen omawiamy szczegółowo w rozdziale 6.

3

Konfigurowanie sprzętu sieciowego



Powiedzieliśmy nieco o interfejsach sieciowych i ogólnie o TCP/IP, ale nie opisaliśmy, co tak naprawdę się dzieje, gdy „kod sieciowy" jądra uzyskuje dostęp do sprzętu. Aby to wyjaśnić, musimy podać trochę informacji o interfejsach i sterownikach.

Na początku jest oczywiście sprzęt, na przykład karta Ethernet, FDDI czy Token Ring: jest to płytka drukowana, wypełniona wieloma małymi układami scalonymi z wypisanymi na nich dziwnymi numerkami, umieszczona w złączu w płycie twojego PC. Nazywamy to ogólnie urządzeniem fizycznym.

Abyś mógł używać karty sieciowej, jądro Linuksa musi zawierać specjalne funkcje, które rozumieją określony dla danego urządzenia sposób dostępu. Oprogramowanie, które implementuje te funkcje, nazywane jest *sterownikiem urządzenia*. Linux ma sterowniki dla wielu różnych typów kart sieciowych: ISA, PCI, MCA, EISA, port równoległy, PCMCIA i najnowszy USB.

Co jednak mamy na myśli, mówiąc, że sterownik „obsługuje" urządzenie? Rozważmy to na przykładzie karty Ethernet. Sterownik powinien komunikować się w jakiś sposób z peryferiami karty: musi wysyłać polecenia i dane do karty, natomiast karta powinna dostarczać wszelkie odebrane dane do sterownika.

W komputerach osobistych IBM komunikacja ta odbywa się przez zestaw adresów wejścia/wyjścia, które są odwzorowywane na rejestry na karcie, a także (lub wyłącznie) przez współdzielony lub bezpośredni dostęp do pamięci. Wszystkie polecenia i dane, jakie jądro wysyła do karty, muszą zostać przesłane na te adresy. Adresy wejścia/wyjścia oraz pamięci są zwykle podawane w postaci adresu początkowego lub *adresu podstawowego* (ang. *base address*). Typowe adresy podstawowe w przypadku kart Ethernet dla magistrali ISA to `0x280` lub `0x300`. Karty przeznaczone dla magistrali PCI mają automatycznie przypisywane własne adresy wejścia/wyjścia.

Zwykle nie musisz się martwić o zagadnienia sprzętowe, takie jak adres podstawowy, ponieważ jądro w czasie startu podejmuje próbę wykrycia lokalizacji karty. Nazywa się to *autowykrywaniem*, co oznacza, że jądro odczytuje kilka lokalizacji pamięci i wejścia / wyjścia oraz porównuje odczytane dane z tym, czego oczekuje, jeżeli dana karta sieciowa jest zainstalowana pod tym adresem. Jednak zdarzają się karty sieciowe, których nie da się wykryć automatycznie. Czasem dzieje się tak w przypadku tanich kart sieciowych, które nie są w pełni klonami standardowych kart innych producentów. W czasie startu jądro próbuje wykryć tylko jedną kartę sieciową. Jeżeli używasz więcej niż jednej karty, musisz jawnie powiedzieć o tym jądru.

Innym parametrem, który być może będzie trzeba podać jądru, jest numer przerwania. Urządzenia zwykle generują przerwanie do jądra, aby na przykład zwrócić na siebie uwagę, gdy nadeszły dane lub wystąpiła jakaś szczególna sytuacja. W komputerach PC z magistralą ISA przerwania mogą pojawiać się na jednym z 15 kanałów przerwań, ponumerowanych następująco: 0, 1, 3 i tak dalej do 15. Numer przerwania przypisany do urządzenia nazywa się *numerem zgłoszenia przerwania* (ang. *Interrupt request number* – IRQ)*.

Z rozdziału 2, *Wybrane problemy sieci TCP/IP*, wiemy, że jądro uzyskuje dostęp do urządzenia sieciowego przez oprogramowanie nazywane *interfejsem*. Interfejsy są zestawami funkcji (np. wysyłania lub odbierania datagramu), identycznymi dla różnych typów urządzeń.

Interfejsy są identyfikowane na podstawie nazw. W wielu uniksowych systemach operacyjnych interfejs sieciowy jest implementowany jako specjalny plik w katalogu */dev*. Jeżeli napiszesz polecenie `ls -las /dev/`, zobaczysz, jak wyglądają takie pliki. Zauważysz, że w kolumnie praw dostępu (drugiej) pliki urządzeń zaczynają się raczej literą, a nie myślnikiem (jak zwykłe pliki). Znak ten określa typ urządzenia. Najpopularniejsze są urządzenia typu `b`, czyli *urządzenia blokowe* obsługujące całe bloki danych przy każdym odczycie i zapisie oraz urządzenia typu `c`, czyli *urządzenia znakowe*, obsługujące dane po jednym znaku. Tam, gdzie zwykle w wyniku pokazywanym przez polecenie *ls* widzisz rozmiar pliku, tu taj są dwie liczby nazywane numerem nadrzędnym i podrzędnym urządzenia. Liczby te wskazują rzeczywiste urządzenie, z którym jest związany plik.

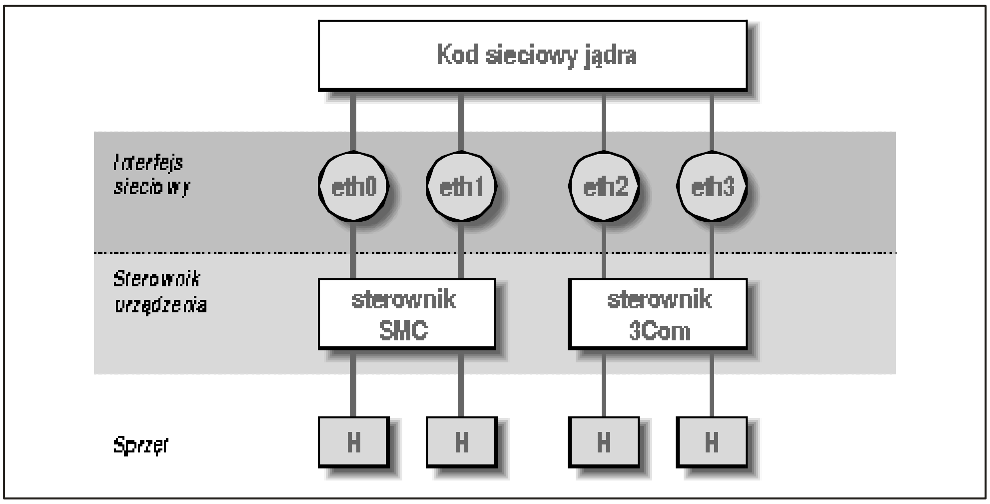
Każdy sterownik rejestruje unikalny numer nadrzędny w jądrze. Każda *instancja* urządzenia rejestruje unikalny numer podrzędny danego urządzenia nadrzędnego. Interfejsy `tty`, */dev/tty* `*`, są urządzeniami znakowymi wskazywanymi przez literę c i każde ma numer nadrzędny `4`, ale */dev/tty1* ma numer podrzędny `1`, a */dev/tty2* ma numer podrzędny 2. Pliki urządzeń są bardzo użyteczne dla wielu typów urządzeń, ale mogą sprawiać kłopoty, gdy chcesz otworzyć nieużywane urządzenie.

Nazwy interfejsów w Linuksie są zdefiniowane wewnętrznie w jądrze i nie są plikami urządzeń w katalogu */dev*. Niektóre typowe nazwy podano w dalszym podrozdziale *Wycieczka po urządzeniach sieciowych Linuksa*. Przypisanie interfejsów do urządzeń zwykle zależy od kolejności, w której są one konfigurowane. Na przykład

* IRQ 2 i 9 są tymi samymi przerwaniami, ponieważ architektura IBM PC posiada dwa kaskadowe procesory po osiem IRQ każdy. Drugi jest połączony z pierwszym poprzez IRQ 2 pierwszego.

pierwsza zainstalowana karta Ethernet będzie nosiła nazwę *eth0*, a następna *eth1*. Interfejsy SLIP są obsługiwane inaczej niż pozostałe urządzenia, ponieważ są przypisywane dynamicznie. Kiedy zostanie zestawione połączenie SLIP, interfejs jest przypisywany do portu szeregowego.

Rysunek 3-1 pokazuje zależności pomiędzy sprzętem, sterownikami urządzenia i interfejsami.



Rysunek 3-1. Związek pomiędzy sterownikami, interfejsami i sprzętem

Przy uruchamianiu systemu jądro wyświetla wykryte urządzenia i instalowane interfejsy. Oto fragment typowych komunikatów wyświetlanych w czasie uruchamiania systemu:

```
.
.   This processor honors the WP bit even when in supervisor mode./
    Good.
Swansea University Computer Society NET3.035 for Linux 2.0
NET3: Unix domain sockets 0.13 for Linux NET3.035.
Swansea University Computer Society TCP/IP for NET3.034
IP Protocols: IGMP, ICMP, UDP, TCP
Swansea University Computer Society IPX 0.34 for NET3.035
IPX Portions Copyright (c) 1995 Caldera, Inc.
Serial driver version 4.13 with no serial options enabled
tty00 at 0x03f8 (irq = 4) is a 16550A
tty01 at 0x02f8 (irq = 3) is a 16550A
CSLIP: code copyright 1989 Regents of the University of California
PPP: Version 2.2.0 (dynamic channel allocation)
PPP Dynamic channel allocation code copyright 1995 Caldera, Inc.
PPP line disciplne registered.
eth0: 3C509 at 0x300 tag 1, 10baseT port, address 00 a0 24 0e e4 e0, /
    IRQ 10.
3c509.c:1.12 6/4/97 becker@cesdis.gsfc.nasa.gov
Linux Version 2.0.32 (root@perf) (gcc Version 2.7.2.1)
#1 Tue Oct 21 15:30:44 EST 1997
 .
 .
```

Ten przykład pokazuje, że jądro zostało skompilowane z włączonym protokołem TCP/IP i zawiera sterowniki dla SLIP, CSLIP i PPP. Trzeci wiersz od końca mówi, że została wykryta karta Ethernet 3C509, która jest zainstalowana jako interfejs *eth0*. Gdybyś miał kartę innego typu, na przykład D-Link pocket adaptor, jądro wypisałoby wiersz rozpoczynający się od nazwy takiego urządzenia – *dl0* w przypadku D-Link, a następnie pokazałoby typ wykrytej karty. Gdybyś miał zainstalowaną kartę sieciową, ale nie widziałbyś żadnego podobnego komunikatu, oznacza to, że jądro nie jest w stanie jej poprawnie wykryć. Sytuacja ta zostanie omówiona w dalszym podrozdziale *Automatyczne wykrywanie kart Ethernet*.

Konfigurowanie jądra

Do wielu dystrybucji Linuksa są dołączane dyskietki startowe, które działają z większością sprzętu PC. Dostarczone jądro jest znacznie zmodularyzowane i zawiera prawie wszelkie możliwe sterowniki. Takie rozwiązanie wygląda świetnie na dyskietce startowej, ale raczej nie przyda się zwykłemu użytkownikowi. Nie ma sensu zajmować miejsca na dysku sterownikami, których nie będziesz używał. Dlatego najlepiej przygotować własne jądro i umieścić w nim tylko te sterowniki, których rzeczywiście potrzebujesz – w ten sposób zaoszczędzisz nieco miejsca na dysku i zmniejszysz czas potrzebny na skompilowanie nowego jądra.

W każdym razie jeżeli pracujesz z Linuksem, powinieneś umieć tworzyć jądro. Uznaj to za potwierdzenie tego, że darmowe oprogramowanie jest świetne – masz kod źródłowy. Nie myśl: „Muszę skompilować jądro”, ale raczej: „*Mogę* skompilować jądro”. Podstawy kompilacji jądra Linuksa zostały wyjaśnione w książce Matta Welsha *Running Linux* (wyd. pol.: *Linux*, Wydawnictwo RM, Warszawa 2000). Dlatego w tym podrozdziale omówimy jedynie opcje konfiguracyjne dotyczące sieci.

Naprawdę ważną rzeczą, którą warto tutaj przypomieć, jest schemat numeracji jądra. Jądra Linuksa są numerowane w formacie: `2.2.14`. Pierwsza cyfra oznacza *główny* numer wersji. Zmienia się ona wtedy, gdy następują poważne, znaczące przekształcenia w architekturze jądra. Na przykład wersję jądra przenumerowano z `1.` na `2.`, gdy zostało dodane wsparcie dla maszyn opartych na nie intelowskich procesorach. Druga liczba to *drugorzędny* numer wersji. Pod wieloma względami ważniejsza jest właśnie ona.

Społeczność twórców Linuksa przyjęła zasadę, że *parzyste* drugorzędne numery wersji oznaczają jądra *produkcyjne* lub *stabilne*, a *nieparzyste* numery wersji oznaczają jądra *rozwojowe* lub *niestabline*. Na maszynie, która jest dla ciebie ważna, powinieneś używać jąder stabilnych, gdyż są one lepiej przetestowane. Po jądra rozwojowe warto sięgnąć wtedy, gdy lubisz eksperymentować z najnowszymi funkcjami Linuksa, ale musisz liczyć się z tym, że mogą pojawić się jeszcze nieznane i niepoprawione błędy. Trzecia liczba to po prostu kolejne wersje wersji oznaczonej numerem drugorzędnym*.

* Powinno się używać jąder rozwojowych i zgłaszać błędy, jeżeli się je znajdzie. Takie eksperymentowanie jest bardzo pouczające, zwłaszcza jeżeli masz komputer, którego możesz używać tylko do testów. Procedura zgłaszania błędów jest szczegółowo podana w pliku */usr/src/linux/REPORTING-BUGS* w kodzie źródłowym jądra Linuksa.

Gdy wy dasz po le ce nie *makemenuconfig*, po ja wi się tek sto we me nu z listą py tań dotyczących konfiguracji. Będą to pytania typu: czy chcesz emulacji koprocesora w jądrze. Jedno z tych py tań do ty czy obsługi sie ci TCP/IP. Mu sisz na nie od powiedzieć y, aby jądro było w sta nie obsłużyć sieć.

Opcje jądra w Linuksie 2.0 i now szych

Po usta le niu ogól ny ch opcji kon fi gu ra cyj nych na stę pują py ta nia o to, czy chcesz zapew nić obsługę różnych funk cji, ta kich jak ste row ni ki SC SI czy kar ty dź wię ko we. Mo nit bę dzie po ka zy wał do stęp ne opcje. Możesz na ci snąć ?, aby za po znać się z opisem da nej opcji. Zaw sze masz do wy bo ru „tak" (y), aby sta tycz nie dołączyć ele ment do jądra, lub „nie" (n), aby usunąć go całko wi cie z jądra. Spo tkasz ta kże opcję modułu (m) w przy pad ku elem ent ów, które mogą zo stać skom pi lo wa ne ja ko mo duły ład o wa ne w cza sie pra cy jądra. Mo duły ład u je się, za nim zostaną wy ko rzy sta ne i są one szczególnie przy dat ne dla sterowników lub rza dziej uży wa nych ele me ntów.

Da lej na stęp uje li sta py tań o obsługę sie ci. Dokładny ze staw opcji kon fig ura cyj nych nie ustann ie się zmie nia ze wzglę du na ciągły rozwój. Ty powa li sta opcji ofer owanych przez wię ks zość jąder ro dziny 2.0 i 2.1 wygląda tak:

```
*
* Network device support
*
Network device support (CONFIG_NETDEVICES) [Y/n/?]
```

Musisz odpowiedzieć na to pytanie y, jeżeli chcesz korzystać z *jakichkolwiek* urządzeń sieciowych, czy to będzie Ethernet, SLIP, PPP, czy cokolwiek innego. Gdy od po wiesz na py ta nie twierdząco, au to ma tycz nie zo sta nie włączo na obsługa urządzeń ty pu Et her net. Je żeli chcesz włączyć obsługę in nych ty pów st erowników sie cio wych, mu sisz od po wie dzieć na do dat ko we py ta nia.

```
PLIP (parallel port) support (CONFIG_PLIP) [N/y/m/?] y
PPP (point-to-point) support (CONFIG_PPP) [N/y/m/?] y
*
* CCP compressors for PPP are only built as modules.
*
SLIP (serial line) support (CONFIG_SLIP) [N/y/m/?] m
 CSLIP compressed headers (CONFIG_SLIP_COMPRESSED) [N/y/?] (NEW) y
 Keepalive and linefill (CONFIG_SLIP_SMART) [N/y/?] (NEW) y
 Six bit SLIP encapsulation (CONFIG_SLIP_MODE_SLIP6) [N/y/?] (NEW) y
```

Py ta nia te do tyczą ró żnych pro to kołów war stwy łącza obsługi wa nych przez Linuksa. Zarówno PPP, jak i SLIP po zwa lają na prze syłanie datagramów IP po łączach sze regowych. PPP w rze czy wi sto ści jest ze sta wem pro to kołów uży wa nych do wy syłania da nych po łączach sze re go wych. Nie kt óre pro to koły wchodzące w skład ze sta wu PPP obsługują uwierzytelnianiesię użytkownika na serwerzedostępowym, natomiast in ne zaj mują się prze no sze niem pew nych pro to kołów przez łącze – PPP transpor tu je nie tyl ko da ta gra my TCP/IP – może ta kże prze no sić in ne pro to koły, ta kie jak IPX.

Je żeli na py ta nie o obsługę pro to kołu SLIP od po wieszy lub m, zosta niesz poproszony o od po wie dź na trzy ko lej ne py ta nia. Opcja kom pre sji nagłów ka po zwa la na ko rzy-

staniez CSLIP – techniki, która kompresuje nagłówki TCP/IP do zaledwie trzech bajtów. Zauważ, że ta opcja jądra nie włącza automatycznie CSLIP, a jedynie udostępnia niezbędne do tego celu funkcje jądra. Opcja `Keepalive and linefill` powoduje, że co jakiś czas jest generowany sztuczny ruch na łączu SLIP, aby uniknąć zerwania połączenia przez czujnik nieaktywności. Opcja `Six bit SLIP encapsulation` pozwala na uruchomienie SLIP na liniach i obwodach, które nie są w stanie przesyłać pełnych 8-bitowych zestawów danych. Jest to technika podobna do uukodowania (ang. *uuencoding*) lub algorytmu binhex, stosowanych do przesyłania plików binarnych pocztą elektroniczną.

Protokół PLIP jest sposobem przesłania datagramów IP przez łącze oparte o porty równoległe. Używa się go do komunikowania się komputerów PC pracujących w systemie DOS. Na typowym komputerze PC, protokół PLIP może być szybszy niż PPP czy SLIP, ale bardziej obciąża procesor, a więc choć przepustowość pozostanie odpowiednia, to inne zadania mogą być realizowane wolniej.

Poniższe pytania dotyczą kart sieciowych różnych sprzedawców. Im więcej sterowników jest dostępnych na rynku, tym bardziej prawdopodobne, że w tej sekcji pojawi się nowe pytanie. Gdybyś chciał stworzyć jądro, mógłbyś robić to na wielu różnych maszynach, a gdyby twoja maszyna miała zainstalowane różne rodzaje kart sieciowych, mógłbyś włączyć więcej niż jeden sterownik:

```
  .
  .
Ethernet (10 or 100Mbit) (CONFIG_NET_ETHERNET) [Y/n/?]
3COM cards (CONFIG_NET_VENDOR_3COM) [Y/n/?]
3c501 support (CONFIG_EL1) [N/y/m/?]
3c503 support (CONFIG_EL2) [N/y/m/?]
3c509/3c579 support (CONFIG_EL3) [N/y/m/?]
3c590/3c900 series (592/595/597/900/905) "Vortex/Boomerang" support
    (CONFIG_VORTEX) [N/y/m/?]
AMD LANCE and PCnet (AT1500 and NE2100) support (CONFIG_LANCE) [N/y/?]
AMD PCInet32 (VLB and PCI) support (CONFIG_LANCE32) [N/y/?] (NEW)
Western Digital/SMC cards (CONFIG_NET_VENDOR_SMC) [N/y/?]
WD80*3 support (CONFIG_WD80x3) [N/y/m/?] (NEW)
SMC Ultra support (CONFIG_ULTRA) [N/y/m/?] (NEW)
SMC Ultra32 support (CONFIG_ULTRA32) [N/y/m/?] (NEW)
SMC 9194 support (CONFIG_SMC9194) [N/y/m/?] (NEW)
Other ISA cards (CONFIG_NET_ISA) [N/y/?]
Cabletron E21xx support (CONFIG_E2100) [N/y/m/?] (NEW)
DEPCA, DE10x, DE200, DE201, DE202, DE422 support (CONFIG_DEPCA) [N/y/m/?] (NEW)
EtherWORKS 3 (DE203, DE204, DE205) support (CONFIG_EWRK3) [N/y/m/?] (NEW)
EtherExpress 16 support (CONFIG_EEXPRESS) [N/y/m/?] (NEW)
HP PCLAN+ (27247B and 27252A) support (CONFIG_HPLAN_PLUS) [N/y/m/?] (NEW)
HP PCLAN (27245 and other 27xxx series) support (CONFIG_HPLAN) [N/y/m/?] (NEW)
HP 10/100VG PCLAN (ISA, EISA, PCI) support (CONFIG_HP100) [N/y/m/?] (NEW)
NE2000/NE1000 support (CONFIG_NE2000) [N/y/m/?] (NEW)
SK_G16 support (CONFIG_SK_G16) [N/y/?] (NEW)
EISA, VLB, PCI and on card controllers (CONFIG_NET_EISA) [N/y/?]
Apricot Xen-II on card ethernet (CONFIG_APRICOT) [N/y/m/?] (NEW)
Intel EtherExpress/Pro 100B support (CONFIG_EEXPRESS_PRO100B) [N/y/m/?] (NEW)
DE425, DE434, DE435, DE450, DE500 support (CONFIG_DE4X5) [N/y/m/?] (NEW)
DECchip Tulip (dc21x4x) PCI support (CONFIG_DEC_ELCP) [N/y/m/?] (NEW)
Digi Intl. RightSwitch SE-X support (CONFIG_DGRS) [N/y/m/?] (NEW)
Pocket and portable adaptors (CONFIG_NET_POCKET) [N/y/?]
```

```
AT-LAN-TEC/RealTek pocket adaptor support (CONFIG_ATP) [N/y/?] (NEW)
D-Link DE600 pocket adaptor support (CONFIG_DE600) [N/y/m/?] (NEW)
D-Link DE620 pocket adaptor support (CONFIG_DE620) [N/y/m/?] (NEW)
Token Ring driver support (CONFIG_TR) [N/y/?]
IBM Tropic chipset based adaptor support (CONFIG_IBMTR) [N/y/m/?] (NEW)
FDDI driver support (CONFIG_FDDI) [N/y/?]
Digital DEFEA and DEFPA adapter support (CONFIG_DEFXX) [N/y/?] (NEW)
ARCnet support (CONFIG_ARCNET) [N/y/m/?]
  Enable arc0e (ARCnet "Ether-Encap" packet format) (CONFIG_ARCNET_ETH)/
      [N/y/?] (NEW)
  Enable arc0s (ARCnet RFC1051 packet format) (CONFIG_ARCNET_1051)/
      [N/y/?] (NEW)
 .
 .
```

Pod koniec sekcji dotyczącej systemu plików skrypt konfiguracyjny zapyta cię, czy chcesz włączyć obsługę NFS – sieciowego systemu plików. NFS pozwala na udostępnienie systemu plików kilku hostom, na których pliki z twojego hosta będą widoczne tak, jak by były na zwykłym dysku podłączonym lokalnie:

```
NFS file system support (CONFIG_NFS_FS) [y]
```

NFS opiszemy szczegółowo w rozdziale 14, *Sieciowy system plików*.

Opcje sieciowe jądra w Linuksie 2.0.0 i nowszych

W jądrze Linuksa 2.0.0 nastąpiły znaczne zmiany w obsłudze sieci. Wiele funkcji stało się standardową częścią jądra, na przykład obsługa protokołu IPX. Dodano także szereg opcji, co otworzyło nowe możliwości konfiguracyjne. Wielu opcji używa się tylko w bardzo szczególnych sytuacjach i nie będziemy ich opisywać. Dokument *Networking-HOWTO* opisuje to, co my tutaj pomijamy. W tym podrozdziale podajemy najbardziej przydatne opcje i wyjaśniamy, kiedy należy ich używać:

Podstawy

Aby używać sieci TCP/IP, musisz odpowiedzieć na to pytanie, wpisując `y`. Jeżeli odpowiesz `n`, wciąż będziesz mógł skompilować jądro z obsługą IPX:

```
Networking options --->
    [*] TCP/IP networking
```

Gatewaye

Musisz włączyć tę opcję, jeżeli twój system pracuje jako gateway pomiędzy dwoma sieciami lub pomiędzy siecią lokalną a łączem SLIP. To, że opcja jest domyślnie włączona, w niczym nie przeszkadza, ale trzeba ją wyłączyć, jeżeli chcesz skonfigurować host jako *firewall*. Firewalle to hosty, które są podłączone do dwóch lub więcej sieci, ale nie rutują ruchu pomiędzy nimi. Są one powszechnie stosowane, aby udostępnić użytkownikom Internet przy minimalnym ryzyku dla sieci wewnętrznej. Użytkownicy mogą logować się do firewalla i korzystać z usług internetowych, a komputery firmowe są zabezpieczone przed atakami z zewnątrz, ponieważ firewalle nie przepuszczają żadnych połączeń przychodzących z zewnątrz (firewalle opisujemy szczegółowo w rozdziale 9, *Firewall TCP/IP*):

```
[*] IP: forwarding/gatewaying
```

Wirtualne hosty

Opcje te pozwalają na skonfigurowanie więcej niż jednego adresu IP dla jednego interfejsu. Przydają się, jeżeli chcesz tworzyć „hosty wirtualne", czyli skonfigurować maszynę tak, że wygląda i działa jak kilka oddzielnych maszyn, każda o własnych parametrach sieciowych. Więcej na temat tworzenia aliasów IP powiemy za chwilę:

```
[*] Network aliasing
   <*> IP: aliasing support
```

Liczenie ruchu IP

Opcja ta pozwala na zbieranie danych na temat wielkości ruchu IP (wychodzącego i wchodzącego) w danej maszynie. (Omówienie tego zagadnienia zawiera rozdział 10, *Liczenie ruchu IP*).

```
[*] IP: accounting
```

Błąd PC

Opcja ta rozwiązuje problem niekompatybilności z niektórymi wersjami zestawu PC/TCP, bedącego komercyjną implementacją TCP/IP dla komputerów PC opartych na DOS-ie. Jeżeli włączysz tę opcję, wciąż będziesz mógł komunikować się ze zwykłymi maszynami uniksowymi, ale wydajność na gorszych łączach może być słabsza:

```
--- (it is safe to leave these untouched)
[*] IP: PC/TCP compatibility mode
```

Uruchamianie bezdyskowe

Funkcja ta włącza RARP (odwrotny protokół rozwiązywania adresów). RARP jest używany przez klienty bezdyskowe i X terminale do uzyskiwania swojego adresu IP przy uruchamianiu. Powinieneś włączyć RARP, jeżeli planujesz obsługę klientów tego typu. Mały program o nazwie *rarp*, dołączony do standardowych narzędzi sieciowych, jest używany to dodawania wpisów do tablicy RARP jądra:

```
<*> IP: Reverse ARP
```

MTU

Aby dane wysyłane przez TCP/IP mogły zostać przekazane do protokołu IP, jądro musi podzielić ich strumień na bloki. Rozmiar bloku jest określany za pomocą *maksymalnej jednostki transmisji* (*Maximum Transmission Unit* – MTU). W przypadku hostów, które są osiągalne przez sieć lokalną, np. Ethernet, typowe jest używanie MTU odpowiadającego maksymalnej wielkości pakietu Ethernet – 1500 bajtom. W przypadku rutingu IP przez sieci rozległe takie jak Internet, preferowane jest stosowanie mniejszych datagramów, aby nie musiały być dalej dzielone w procesie zwanym *fragmentacją IP* (ang. *IP fragmentation*)*. Jądro jest w stanie automatycznie określić najmniejszą wartość MTU dla danej trasy IP i automatycznie skonfigurować połączenie TCP dla tej trasy. Zachowanie to jest do-

* Pamiętaj, że protokół IP może być przesyłany przez różne typy sieci, a nie wszystkie obsługują tak duży rozmiar pakietu jak sieć Ethernet.

myślne. Jeżeli odpowiesz `y` przy wyborze tej opcji, właściwość ta zostanie wyłączona.

Jeżeli rzeczywiście chcesz wysyłać dane w mniejszych pakietach do określonych hostów (poniew aż na przykład dane są przesyłane przez łącze SLIP), skorzystaj z opcji *mss* polecenia *route*, które zostanie krótko omówione pod koniec tego rozdziału:

```
[ ] IP: Disable Path MTU Discovery (normally enabled)
```

Bezpieczeństwo

Protokół IP obsługuje funkcję *źródłowego wyboru trasy* (ang. *source routing*). · ródłowy wybór trasy pozwala na zakodowanie trasy datagramu w nim samym. Z całą pewnością było to dobre rozwiązanie, zanim upowszechniły się protokoły RIP i OSPF. Obecnie uznawane jest za niebezpieczne, ponieważ daje osobom niepowołanym narzędzie do pokonania zabezpieczeń firewalli, mianowicie pozwala na ominięcie tablicy rutingu rutera. W normalnej sytuacji powinieneś filtrować datagramy ze źródłowym wyborem trasy, a więc ta opcja powinna być włączona:

```
[*] IP: Drop source routed frames
```

Obsługa sieci Novell

Opcja ta włącza obsługę IPX – protokołu transportowego wykorzystywanego w sieci Novell. Linux będzie działał doskonale jako ruter IPX, a ponadto funkcja ta jest przydatna w środowiskach, gdzie znajdują się serwery plików Novell. System plików NCP również wymaga włączonej obsługi IPX w jądrze. Gdybyś chciał podłączyć się i zamontować systemy plików Novell, musiałbyś mieć tę opcję włączoną (IPX i system plików NCP omawiamy w rozdziale 15, *IPX i system plików NCP*):

```
<*> The IPX protocol
```

Radioamatorskie

Trzy poniższe opcje włączają obsługę protokołów radioamatorskiego obsługiwanych przez Linuksa: AX.25, NetRom i Rose (nie opisujemy ich w tej książce, ale są one szczegółowo przedstawione w dokumencie *AX25-HOWTO*):

```
<*> Amateur Radio AX.25 Level 2
<*> Amateur Radio NET/ROM
<*> Amateur Radio X.25 PLP (Rose)
```

Linux obsługuje jeszcze jeden typ sterownika: sterownik fikcyjny (ang. *dummy driver*). Poniższe pytanie pojawia się na początku sekcji dotyczącej sterowników urządzeń:

```
<*> Dummy net driver support
```

Sterownik fikcyjny jest w zasadzie przydatny tylko w przypadku samodzielnych hostów PPP/SLIP. Jest to w istocie interfejs pętli zwrotnej z maskowaniem IP. Na hostach, które posiadają jedynie interfejsy PPP/SLIP, będziesz chciał mieć interfejs, który przez cały czas utrzymuje twój adres IP. Omawiamy to nieco bardziej szczegółowo w podrozdziale *Interfejs fikcyjny* w rozdziale 5, *Konfigurowanie sieci TCP/IP*.

Zauważ, że dzisiaj to samo możesz uzyskać, używając aliasu IP i konfigurując swój adres IP jako alias na interfejsie pętli zwrotnej.

Wycieczka po urządzeniach sieciowych Linuksa

Jądro Linuksa obsługuje szereg sterowników dla różnego rodzaju sprzętu. Ten podrozdział to krótki przegląd dostępnych rodzin sterowników i używanych przez nie nazw interfejsów.

Interfejsy w Linuksie mają standardowe nazwy, wymienione poniżej. Większość sterowników obsługuje więcej niż jeden interfejs, dlatego interfejsy są numerowane, na przykład *eth0* i *eth1*.

lo

To lokalny interfejs pętli zwrotnej. Jest używany zarówno do celów testowych, jak i przez kilka aplikacji sieciowych. Działa na zasadzie obwodu zamkniętego, w którym wszelkie dane wysłane do interfejsu są zwracane do warstwy sieciowej hosta. W jądrze istnieje zawsze tylko jeden interfejs pętli zwrotnej i nie ma sensu, aby było ich więcej.

eth0, eth1...

To interfejsy kart Ethernet. Są używane przez większość kart Ethernet, włącznie z tymi podłączanymi przez port równoległy.

tr0, tr1...

To interfejsy kart Token Ring. Są używane przez większość kart Token Ring, włącznie z produkowanymi przez firmy inne niż IBM.

sl0, sl1...

To interfejsy SLIP. Są związane z łączami szeregowymi w kolejności alokowania dla SLIP.

ppp0, ppp1...

To interfejsy PPP. Podobnie jak interfejsy SLIP, interfejs PPP jest związany z łączem szeregowym pracującym w trybie PPP.

plip0, plip1...

To interfejsy PLIP. PLIP przesyła datagramy IP przez łącza równoległe. Interfejsy są alokowane przez sterownik PLIP w czasie uruchamiania systemu i są odwzorowane na porty równoległe. W jądrach *2.0.x* istnieje bezpośredni związek między nazwą urządzenia a portem wejścia/wyjścia portu równoległego, ale w nowszych jądrach nazwy urządzeń są przypisywane kolejno, tak jak w urządzeniach SLIP i PPP.

ax0, ax1...

To interfejsy AX.25. AX.25 jest podstawowym protokołem używanym przez operatorów radia amatorskiego. Interfejsy AX.25 są alokowane i przypisywane w podobny sposób jak urządzenia SLIP.

Istnieje wiele innych typów interfejsów dla innych urządzeń sieciowych. Wymieniliśmy tylko najpopularniejsze z nich.

W kilku następnych podrozdziałach omówimy dokładniej korzystanie z opisanych powyżej sterowników. Dokument *Networkig-HOWTO* opisuje konfigurację większości pozostałych interfejsów, natomiast *AX25-HOWTO* wyjaśnia, jak skonfigurować urządzenia sieciowe radia amatorskiego.

Instalowanie Ethernetu

Kod sieciowy Liunksa w obecnej postaci obsługuje wiele kart Ehternet. Większość sterowników została napisana przez Donalda Beckera, który stworzył rodzinę sterowników dla kart opartych o układ NationalSemiconductor 8390. Są one znane pod nazwą Becker Series Drivers. Sterowniki dla różnego sprzętu pisali też inni programiści. Dzięki temu większość popularnych kart jest obsługiwana przez Linuksa, z naprawdę nielicznymi wyjątkami. Lista obsługiwanych kart Ethernet stale się wydłuża, a więc jeżeli twoja karta jeszcze się na niej nie znajduje, to istnieje realna szansa, że wkrótce tam dołączy.

Niegdyś próbowano sporządzić listę wszystkich obsługiwanych kart Ethernet, ale obecnie zajęłoby to zbyt dużo czasu i miejsca. Na szczęście Paul Gortmaker, który redaguje dokument *Ethernet-HOWTO*, zamieszcza listę wszystkich obsługiwanych kart i podaje przydatne informacje na temat ich uruchamiania w Linuksie*. Co miesiąc jest ona wysyłana do grupy dyskusyjnej *comp.os.linux.answers*, a także jest dostępna w ośrodkach lustrzanych Projektu Dokumentacji Linuksa.

Nawet, jeżeli jesteś przekonany, że potrafisz zainstalować dany typ karty Ethernet w swoim komputerze, warto zajrzeć do *Ethernet-HOWTO* i dowiedzieć się, co ma do powiedzenia na ten temat. Znajdziesz tam informacje wykraczające poza proste zagadnienia konfiguracji. Na przykład zapewne unikniesz niepotrzebnych kłopotów, jeśli będziesz wiedział, jak się zachowują niektóre karty Ethernet oparte na DMA i wykorzystujące ten sam kanał DMA, który jest domyślnie przeznaczony dla kontrolera SCSI Adaptec AHA 1542. Dopóki nie przełączysz ich na inny kanał DMA, uruchomienie komputera będzie się kończyło zapisywaniem pakietów przez kartę Ethernet na losowe miejsca twojego dysku twardego.

Aby skorzystać z dowolnej obsługiwanej przez Linuksa karty Ethernet, możesz użyć prekompilowanego jądra z jakiejś znanej dystrybucji Linuksa. Zwykle mają one moduły dla wszystkich obsługiwanych sterowników, a w procesie instalacji zwykle możesz wybrać te sterowniki, które chcesz załadować. Jednak na dłuższą metę lepiej jest skompilować własne jądro i umieścić w nim tylko te sterowniki, które są rzeczywiście potrzebne. Zaoszczędzisz miejsce na dysku i pamięć.

Automatyczne wykrywanie kart Ethernet

Sterowniki Ethernet w Linuksie są zwykle na tyle inteligentne, by znaleźć lokalizację karty Ethernet. Dzięki temu nie musisz sam wskazywać jej jądra. *Ethernet-HOWTO* informuje, czy dany sterownik używa automatycznego wykrywania i w jakiej kolejności sprawdza adresy wejścia/wyjścia karty.

---

* Z Paulem można się skontaktować pod adresem *gpg109@rsphy1.anu.edu.au*.

Kod automatycznego wykrywania ma trzy ograniczenia. Po pierwsze, nie jest on w stanie poprawnie rozpoznać wszystkich kart. Jest to szczególnie widoczne w przypadku tańszych klonów popularnych kart. Po drugie, jądro nie wykryje automatycznie więcej niż jednej karty, dopóki mu tego jawnie nie zaznaczysz. Jest to świadome założenie konstrukcyjne, gdyż uznano, że będziesz chciał mieć kontrolę nad tym, która karta jest przypisywana do którego interfejsu. Najlepszym sposobem na zrobienie tego porządnie jest ręczne skonfigurowanie kart Ethernet we własnym komputerze. Po trzecie, sterownik może przeoczyć adres, pod którym jest skonfigurowana twoja karta. Podsumowując, sterowniki będą automatycznie szukały karty tylko pod tymi adresami, pod którymi dane urządzenie może być skonfigurowane, ale czasem pewne adresy są ignorowane w celu uniknięcia konfliktów sprzętowych z innymi typami kart, które często wykorzystują ten sam adres.

Karty sieciowe PCI powinny być wykrywane bez kłopotów. Jeżeli jednak używasz więcej niż jednej karty albo jeżeli automatyczne wykrywanie się nie powiedzie, istnieje sposób na jawne powiadomienie jądra o adresie podstawowym i nazwie karty.

W czasie uruchamiania systemu możesz podać do jądra argumenty i informacje, które mogą się przydać niektórym jego składnikom. Mechanizm ten pozwala ci na przykład na przekazanie do jądra informacji, które umożliwią sterownikom Ethernet zlokalizowanie sprzętu Ethernet bez wykrywania go przez sterownik.

Jeżeli korzystasz z systemu uruchamiania *lilo*, możesz przekazać parametry do jądra, wpisując je za pomocą opcji `append` w pliku *lilo.conf*. Aby powiadomić jądro o urządzeniu Ethernet, możesz przekazać mu następujące parametry:

```
ether=irq,base_addr,[param1,][param2,]name
```

Pierwsze cztery parametry są liczbami, natomiast ostatni to nazwa urządzenia. Obowiązkowe są *`irq`*, *`base_addr`* i *`name`*, opcjonalne – dwa parametry *`param`*. Dowolne wartości liczbowe mogą być ustawione na zero, co powoduje, że jądro określi je przez wykrywanie.

Pierwszy parametr określa IRQ przypisane do urządzenia. Domyślnie jądro będzie próbowało automatycznie wykryć kanał IRQ urządzenia. Sterownik 3c503, na przykład, ma specjalną funkcję, która wybiera wolne IRQ z listy 5, 9, 3, 4 i konfiguruje kartę tak, by z niego korzystała. Parametr *`base_addr`* określa podstawowy adres wejścia/wyjścia karty – wartość zero mówi jądru, by sprawdziło podane adresy.

Kolejne dwa parametry są różnie wykorzystywane przez różne sterowniki. W przypadku kart wykorzystujących współdzieloną pamięć, takich jak WD80x3, parametry te określają adresy początkowy i końcowy obszaru pamięci. Inne karty powszechnie używają *`param1`* do ustawienia poziomu wyświetlanych informacji debugujących. Wartości od 1 do 7 wyznaczają kolejne poziomy ilości informacji, natomiast 8 wyłącza je wszystkie. 0 jest wartością domyślną. Sterownik 3c503 używa *`param2`* do wyboru pomiędzy wewnętrznym (domyślnie) a zewnętrznym (wartość 1) transceiverem. Ten pierwszy wykorzystuje złącze karty BNC, natomiast drugi jej port AUI. Argumenty *`param`* nie muszą być w ogóle podawane, jeżeli nie masz nic szczególnego do skonfigurowania.

Pierwszy, nieliczbowy argument jest interpretowany przez jądro jako nazwa urządzenia. Musisz podać nazwę urządzenia dla każdej konfigurowanej karty Ethernet.

Gdybyś miał dwie karty Ethernet, Linux mógłby wykryć jedną kartę automatycznie i przez *lilo* przekazać parametry do drugiej karty, ale prawdopodobnie wolałbyś ręcznie skonfigurować obie karty. Jeśli decydujesz się na wykrywanie jednej karty przez jądro i ręczne konfigurowanie drugiej, musisz mieć pewność, że jądro nie znajdzie przypadkowo najpierw drugiej karty i że pierwsza zostanie w ogóle znaleziona. Dlatego przekaż do *lilo* opcję `reserve`, która jawnie mówi jądru, by nie sprawdzało obszaru wejścia/wyjścia zajętego przez drugą kartę. Na przykład, aby Linux zainstalował drugą kartę Ethernet znajdującą się pod adresem `0x300` jako *eth1*, musiałbyś przekazać jądru następujące parametry:

```
reserve=0x300,32 ether=0,0x300,eth1
```

Opcja `reserve` gwarantuje, że żaden sterownik nie będzie miał dostępu do obszaru wejścia/wyjścia drugiej karty w czasie wykrywania innych urządzeń. Możesz także użyć parametru jądra, który unieważnia automatyczne wykrywanie *eth0*:

```
reserve=0x340,32 ether=0,0x340,eth0
```

Możesz także w ogóle wyłączyć automatyczne wykrywanie, na przykład, aby jądro nie próbowało szukać karty Ethernet, którą tymczasowo usunąłeś. W tym celu ustaw argument *`base_addr`* na wartość -1:

```
ether=0,-1,eth0
```

Aby przekazać te parametry do jądra w czasie uruchamiania, wpisujesz je w monicie „`boot:`" *lilo*. Aby *lilo* pokazało monit „`boot:`", musisz nacisnąć jeden z klawiszy [Control], [Alt] lub [Shift] w czasie uruchamiania *lilo*. Jeżeli mając monit, naciśniesz klawisz [Tab], pojawi się lista jąder. Aby uruchomić jądro z podanymi parametrami, wprowadź nazwę wybranego jądra, a następnie spację i parametry, które chcesz przekazać. Po naciśnięciu [Enter] *lilo* załaduje jądro z uwzględnieniem podanych parametrów.

Aby te nowe parametry pojawiły się automatycznie przy ponownym uruchamianiu systemu, wprowadź je do pliku */etc/lilo.conf*, używając słowa kluczowego `append=`. Oto przykład:

```
boot=/dev/hda
root=/dev/hda2
install=/boot/boot.b
map=/boot/map
vga=normal
delay=20
append="ether=10,300,eth0"

image=/boot/vmlinuz-2.2.14
label=2.2.14
read-only
```

Po edycji pliku *lilo.conf* musisz ponownie uruchomić polecenie *lilo*, aby uaktywnić zmiany.

Sterownik PLIP

Prot okół *IP łącza równoległego* (*Parallel Line IP* – PLIP) to łatwy i tani sposób na połącze nie dwó ch ma szyn w sieć. Wy ko rzy stu je port równoległy i spe cjal ny ka bel. Osiąga pręd kość od 10 do 20 kilobajtów na se kun dę.

PLIP po wstał w fir mie Cyrnwr, Inc. Na swo je cza sy od zna czał się po mysłową (lub, je śli wo lisz, ty po wo ha kerską ar chi tek turą), po nie waż ory gi nal ne por ty rów noległe IBM PC były projek to wa ne ja ko jed no kie run ko we por ty dru kar ki. Osiem li nii da nych służyło do wy syłania da nych je dy nie z PC do urządze nia pe ry fe ryj ne go, ale nie w drugą stro nę.* Prot okół PLIP fir my Cyrnwr zno sił to ogra ni cze nie. W PLIP do przyjmowania danych przeznaczono tylko pięć linii stanu portu, co ograniczyło wielkoć dostarczanych danych do półbajtu, ale dopuszczono przesyłanie w obie stro ny. Ten tryb działania zo stał na zwa ny PLIP tryb 0. Obec nie por ty rów noległe PC obsługują pełne dwu kie run ko we prze syłanie da nych 8-b it owych, a PLIP zo stał roz sze rzo ny i no si na zwę PLIP tryb 1.

Jądra Linuk sa do wer sji 2.0 (włącznie) obsługi wały je dy nie PLIP tryb 0, ale ist niały roz sze rzo ne ste row ni ki por tu rów noległego (w po sta ci po pra wek dla jądra 2.0 i ja ko standardowy kod w jądrze 2.2), które obsługiwały także PLIP tryb 1**. W od - różnieniu od wcze śniej szych wer sji ko du PLIP, obec ny ste row nik pró buje być kom pa ty bil ny z im ple men ta cja mi PLIP fir my Cyrnwr oraz ste row ni kiem PLIP umiesz czo nym w NC SA *telnet****. Aby połączyć dwa kom pu te ry za po mocą PLIP, mu sisz mieć spe cjal ny ka bel sprze da wa ny w niekt ór ych skle pach pod nazwą Null Prin ter lub Tur bo La plink. Możesz jed nak wy ko nać go sa mo dziel nie i nie jest to trud ne. Do da tek B, *Przy dat ne kon fi gu ra cje ka bli*, wy ja śnia, jak to zro bić.

Ste row nik PLIP dla Linuk sa jest dziełem pra wie nie zli czo nej rze szy uży tkowników. Obec nie znaj du je się pod opieką Nii be Yu ta ka (ad res kon tak to wy: *gniibe@mri.co.jp*). Sterownik po wkompilowaniu w jądro kon fi gu ru je in ter fejs sie cio wy dla ka żde go możliwego portu drukarki, gdzie *plip0* odpowiada por to wi *lp0*, *plip1* por to wi *lp1* i tak dalej. Od wzo ro wa nie in ter fej sów na por ty in a czej wygląda w jądrach 2.0 niż w jądrach 2.2. W jądrach 2.0 odwzorowanie było zdefiniowane w pli ku *drives/ net/Space.c* w ko dzie jądra i nie mogło się zmie nić. Do my śl ne od wzo ro wa nie w tym pli ku jest na stę pujące:

---

* Walcz o oczysz cze nie z zarzut ów na zwy ha ker! Zaw sze uży waj na zwy „cra ker", gdy mów isz o lu dziach, kt órzy pr óbują po ko nać sys tem za bez pie czeń, a „ha ker", gdy mó wisz o lu dziach, kt órzy wy my śli li mądry sp os ób na roz wiąza nie pro ble mu. Ha ke rzy mogą być cra ke ra mi, ale nie na l e ży ich nig dy ze sobą my lić. Zaj rzyj do *Nowego słownika Hakerów* (New Hac kers Dic tio na ry), któ ry mo żna zna le źć w po sta ci pli ku *Jargon*, a le piej zro zu miesz te po ję cia.

** Poprawka obsługująca rozszerzony port równoległy w jądrach 2.0 jest dostępna pod adresem *http://www.cyberelk.demon.co.uk/parport.html*.

*** NCSA *telnet* to po pu lar ny pro gram dla DO S-a, kt óry po zwa la na uży wa nie TCP/IP w sie ci Et her net lub PLIP i obsługujący usługi *telnet* oraz FTP.

Interfejs	*Port wejścia/wyjścia*	*IRQ*
plip0	`0x3BC`	7
plip1	`0x378`	7
plip2	`0x278`	5

Gdybyś skonfigurował swój port drukarki w inny sposób, musiałbyś zmienić odpowiednie wartości w pliku *drivers/net/Space.c* w kodzie źródłowym jądra Linuksa, które trzeba byłoby przekompilować.

W jądrach 2.2 sterownik PLIP wykorzystuje sterownik portu równoległego „parport" napisany przez Philipa Blundella*. Nowy sterownik przypisuje nazwy urządzeń sieciowych PLIP kolejno, tak jak sterowniki Ethernet czy PPP, a więc pierwsze utworzone urządzenie PLIP ma nazwę *plip0*, drugie *plip1* i tak dalej. Fizyczne porty równoległe są również przypisywane kolejno. Domyślnie sterownik portu równoległego zastosuje procedurę automatycznego wykrywania, aby zidentyfikować sprzęt, który go obsługuje, i kolejno zapisze uzyskiwane informacje o urządzeniu fizycznym. Lepiej jest jawnie przekazać jądru fizyczne parametry wejścia/wyjścia. W tym celu trzeba podać argumenty do modułu *parport_pc.o* w czasie jego ładowania, a jeżeli sterownik jest wkompilowany w jądro, argumenty podaje się w czasie uruchamiania *lilo*. Ustawienia IRQ dowolnego urządzenia mogą zostać zmienione później przez zapisanie nowej wartości IRQ do pliku */proc/parport/*/irq*.

Konfigurowanie parametrów fizycznych wejścia/wyjścia w jądrze 2.2 w czasie ładowania modułu jest proste. Na przykład, aby przekazać sterownikowi, że masz dwa porty równoległe typu PC pod adresami wejścia/wyjścia `0x278` i `0x378` oraz IRQ odpowiednio 5 i 7, możesz załadować moduł z następującymi argumentami:

```
modprobe parport_pc io=0x278,0x378 irq=5,7
```

Odpowiednie argumenty przekazywane do jądra w przypadku wkompilowanego sterownika są następujące:

```
parport=0x278,5 parport=0x378,7
```

Aby argumenty te przekazać do jądra automatycznie w czasie uruchamiania systemu, musisz użyć słowa kluczowego `append` w *lilo*.

Gdy sterownik PLIP zostanie zainicjowany, czy to w czasie uruchamiania systemu, jeżeli jest wbudowany, czy też w czasie ładowania modułu *plip.o*, każdy z portów równoległych będzie miał związane z nim urządzenie sieciowe *plip*. Urządzenie *plip0* zostanie przypisane do pierwszego portu równoległego, *plip1* do drugiego i tak dalej. To przypisanie można pominąć, ręcznie zadając inny zestaw argumentów jądra. Na przykład, aby przypisać `parport0` do urządzenia `plip0` i `parport1` do urządzenia `plip1`, użyłbyś następujących argumentów jądra:

```
plip=parport1 plip=parport0
```

Jednak takie przypisanie nie znaczy, że nie możesz wykorzystywać tych portów równoległych do drukowania czy innych celów. Fizyczne porty równoległe są uży-

* Z Philipem możesz skontaktować się, pisząc na adres *Philip.Blundell@pobox.com*.

wane przez sterownik PLIP jedynie wtedy, gdy odpowiadający im interfejs jest w trybie `up`.

Sterowniki PPP i SLIP

Protokoły PPP (*Point-to-point Protocol* – protokół punkt-punkt) i SLIP (*Serial Line IP* – IP łącza szeregowego) są powszechnie stosowane do przesyłania pakietów IP przez łącze szeregowe. Wiele firm oferuje dostęp komutowany PPP i SLIP do maszyn, które są podłączone do Internetu, zapewniając w ten sposób połączenia IP dla prywatnych osób (często inaczej trudno dostępne).

Aby uruchomić PPP czy SLIP, nie trzeba modyfikować sprzętu – możesz użyć dowolnego portu szeregowego. Ponieważ konfiguracja portu szeregowego nie jest istotą sieci TCP/IP, zagadnienie to znalazło się w rozdziale 4, *Konfigurowanie urządzeń szeregowych*. Natomiast PPP omawiamy szczegółowo w rozdziale 8, *Protokół punkt-punkt*, a SLIP - w rozdziale 7, *IP łącza szeregowego*.

Inne typy sieci

Większość pozostałych typów sieci jest konfigurowana podobnie jak Ethernet. Argumenty przekazywane do modułów ładowalnych będą oczywiście inne, a niektóre sterowniki mogą obsługiwać tylko jedną kartę, ale cała reszta jest taka sama. Dokumentację tych kart możesz znaleźć w katalogu */usr/src/linux/Documentation/networking* w kodzie źródłowym Linuksa.

4

Konfigurowanie urządzeń szeregowych



Internet rozwija się bardzo szybko. A przecież większość jego użytkowników stanowią ci, którzy nie mogą sobie pozwolić na stałe i szybkie łącza i używają protokołów takich jak SLIP, PPP czy UUCP, dzwoniąc do dostawcy usług internetowych i odbierając dzienną porcję swojej poczty i wiadomości grup dyskusyjnych.

Rozdział niniejszy ma pomóc tym wszystkim, którzy swoje połączenie ze światem zewnętrznym opierają na modemach. Nie będziemy mówili, jak skonfigurować modem (instrukcja konkretnego urządzenia powie ci więcej na ten temat), ale opiszemy aspekty specyficzne dla Linuksa i dotyczące zarządzania urządzeniami wykorzystującymi porty szeregowe. Interesujące nas tematy to: oprogramowanie do komunikacji szeregowej, tworzenie plików urządzeń szeregowych, urządzenia szeregowe i konfigurowanie urządzeń szeregowych za pomocą poleceń *setserial* i *stty*. Wiele innych tematów można znaleźć w *Serial-HOWTO* autorstwa Davida Lawyera*.

Oprogramowanie komunikacyjne do połączeń modemowych

Istnieje wiele pakietów komunikacyjnych dla Linuksa. Głównie są to *programy terminala*, które pozwalają użytkownikowi dzwonić do innego komputera i poczuć się tak, jakby siedział przed prostym terminalem. Tradycyjny program terminala dla środowisk uniksowych to *kermit*. Obecnie jest on już nieco przestarzały i może wydawać się trudny. Istnieją wygodniejsze programy, które obsługują funkcje, takie jak książki telefoniczne, języki skryptowe do automatycznego dzwonienia i logowania się do zdalnych systemów komputerowych oraz różne protokoły wymiany plików. Jednym z tych programów jest *minicom*, wzorowany na najpopularniejszym

* Z Davidem można skontaktować się pod adresem *bf347@lafn.org*.

DOS-owym programie terminala. Użytkownicy X11 także mają narzędzie dla siebie – *seyon* jest w pełni funkcjonalnym programem komunikacyjnym opartym na X11.

Programy terminala nie są jedynym rodzajem programów do połączeń szeregowych. Inne pozwalają połączyć się z hostem i pobrać wiadomości grup dyskusyjnych oraz pocztę w jednej paczce, aby później, w wolnej chwili, zapoznać się z nimi i dać odpowiedź. Może zaoszczędzisz w ten sposób dużo czasu i pieniędzy, jeżeli złożyło się tak nieszczęśliwie, że mieszkasz w rejonie, gdzie połączenia lokalne są płatne*. Podczas czytania i przygotowania odpowiedzi nie musisz mieć połączenia z siecią, a gdy będziesz gotowy, zadzwonisz ponownie i umieścisz swoje odpowiedzi na serwerze za jednym zamachem. Potrzebujesz też nieco więcej miejsca na dysku twardym, ponieważ wszystkie wiadomości muszą być na nim umieszczone, zanim je przeczytasz, ale może być to sensowny kompromis przy obecnych cenach dysków twardych.

UUCP zawiera w sobie właśnie tego typu oprogramowanie komunikacyjne. Jest to zestaw programów, które kopiują pliki z jednego hosta na drugi i uruchamiają programy na hoście zdalnym. Często jest używany do przenoszenia poczty czy grup dyskusyjnych w sieciach prywatnych. Pakiet UUCP Ia na Taylora, działający także pod Linuksem, opisujemy szczegółowo w rozdziale 16, *Zarządzanie UUCP Taylora*. Pozostałe nieinteraktywne oprogramowanie komunikacyjne jest używane w sieciach takich, jak Fidonet. Wersje aplikacji pochodzące z Fidonet, takie jak *ifmail*, są również dostępne, chociaż wydaje się nam, że już niewiele osób z nich korzysta.

PPP i SLIP są pośrodku, gdyż pozwalają zarówno na interaktywne, jak i nieinteraktywne użycie. Wiele osób używa PPP i SLIP w celu dzwonienia do swoich sieci campusowych lub innych dostawców Internetu, a potem korzysta z FTP lub czyta strony WWW. PPP i SLIP są także powszechnie stosowane do połączeń stałych i półstałych pomiędzy sieciami LAN, choć jest to ciekawe jedynie przy połączeniach ISDN lub innych o podobnej szybkości.

Wprowadzenie do urządzeń szeregowych

Jądro Linuksa daje możliwość komunikacji z urządzeniami szeregowymi, zwykle nazywanymi urządzeniami *tty*. Jest to skrót od angielskiej nazwy *Teletype device***, która wskazuje na głównego producenta urządzeń terminalowych z początków Uniksa. Termin „urządzenie tty" odnosi się obecnie do wszelkich terminali znakowych. W tym rozdziale zawężamy jego zakres wyłącznie do plików urządzeń w Linuksie, czyli oznacza on tutaj fizyczny terminal.

Linux udostępnia trzy klasy urządzeń tty: urządzenia szeregowe, terminale wirtualne (do których masz dostęp przez naciśnięcie klawiszy od [Alt+F1] do [Alt+F*nn*] na konsoli lokalnej) i pseudoterminale (podobne do potoków dwukierunkowych i używane przez aplikacje takie jak X11). Te pierwsze zostały nazwane urządzeniami tty,

* W USA rozmowy lokalne są przeważnie bezpłatne (–przyp. tłum.).

** *teletype* to po polsku „dalekopis", ale w tym przypadku chodzi o firmę o identycznie brzmiącej nazwie (–przyp. tłum.).

ponieważ oryginalne terminale znakowe były podłączone do maszyny unikswej przez kabel szeregowy lub linię telefoniczną i modem. Dwa kolejne zostały też zaliczone do grupy urządzeń tty, ponieważ z punktu widzenia programisty działały podobnie do tych pierwszych.

SLIP i PPP są przeważnie implementowane w jądrze. Jądro w rzeczywistości nie traktuje urządzenia tty jako urządzenia sieciowego, którym możesz posługiwać się tak jak urządzeniem Ethernet, używając polecenia *ifconfig*. Natomiast widzi je jako coś, do czego może podłączyć urządzenia sieciowe. Aby to zrobić, jądro zmienia tzw. protokół obsługi (ang. *line discipline*) urządzenia tty. Zarówno SLIP, jak i PPP są protokołami obsługi, które mogą zostać włączone na urządzeniach tty. Ogólna zasada jest taka, że sterownik szeregowy obsługuje otrzymane dane w różny sposób, w zależności od tego, z jakiego protokołu obsługi korzysta. W przypadku domyślnego protokołu obsługi sterownik po prostu przesyła kolejno każdy otrzymany znak. Gdy zostanie wybrany protokół obsługi PPP lub SLIP, sterownik nie czyta bloków danych, ale opatruje je specjalnym nagłówkiem, który pozwala na identyfikację bloku danych w strumieniu po drugiej stronie i przesłanie nowego bloku danych. Na razie zrozumienie tego nie jest zbyt istotne. W dalszych rozdziałach omówimy dokładniej PPP oraz SLIP i wszystko stanie się jasne.

Dostęp do urządzeń szeregowych

Tak jak wszystkie urządzenia w systemie Unix, tak i porty szeregowe są dostępne poprzez specjalne pliki urządzeń znajdujące się w katalogu */dev*. Istnieją dwa rodzaje plików urządzeń związanych ze sterownikami szeregowymi i dla każdego portu istnieje jeden taki plik. Zachowanie urządzenia będzie zależało od tego, który z jego plików otworzymy. Tutaj wskażemy różnice, co pomoże nam zrozumieć pewne konfiguracje. Jednak w praktyce wystarczy używać tylko jednego z tych plików. Niedługo jeden z nich może w ogóle przestać istnieć.

Ważniejsze urządzenia z dwóch klas urządzeń szeregowych mają numer nadrzędny 4, a pliki specjalne urządzeń noszą nazwy *ttyS0*, *ttyS1* itd. Drugi rodzaj ma numer nadrzędny 5 i został stworzony do wykorzystania przy dzwonieniu na zewnątrz przez port. Pliki specjalne w tym przypadku noszą nazwy *cua0*, *cua1* itd. W świecie Uniksa liczenie generalnie rozpoczyna się od zera, choć inteligentni ludzie zwykle liczą od jednego. Wprowadza to lekkie zamieszanie, ponieważ `COM1:` jest reprezentowany przez */dev/ttyS0*, `COM2:` przez */dev/ttyS1* itd. Każdy, kto zna architekturę sprzętową IBM PC wie, że port `COM3:` i porty o większych numerach nigdy nie stały się w rzeczywistości standardem.

Urządzenia *cua*, czyli „służące do dzwonienia" (z ang. *calling out*), miały rozwiązać problem konfliktów urządzeń szeregowych przeznaczonych dla modemów i obsługujących zarówno połączenia przychodzące, jak i wychodzące. Niestety, stały się one źródłem innych kłopotów i zapewne trzeba będzie z nich zrezygnować. Przyjrzyjmy się pokrótce problemowi.

Linux, podobnie jak Unix, pozwala, by urządzenie lub inny plik były otwierane przez więcej niż jeden proces jednocześnie. Niestety nie jest to zaletą w przypadku

urządzeń tty, gdyż dwa procesy prawie na pewno będą sobie przeszkadzały. Na szczęście wymyślono mechanizm pozwalający sprawdzać procesowi, czy urządzenie tty zostało już otwarte przez inny proces. Mechanizm ten wykorzystuje tak zwane *pliki blokujące* (ang. *lock files*). Działa na następującej zasadzie: gdy proces chce otworzyć urządzenie tty, sprawdza, czy w określonym miejscu istnieje plik o nazwie podobnej do urządzenia, które chce otworzyć. Jeżeli plik nie istnieje, proces go tworzy i otwiera urządzenie tty. Jeżeli plik istnieje, proces zakłada, że urządzenie otworzył już inny proces i podejmuje stosowne działanie. Jeszcze jeden pomysł na sprawne działanie systemu zarządzania plikami blokującymi to zapisywanie w samym pliku ID procesu (pid), który stworzył plik blokujący. Więcej na ten temat powiemy za chwilę.

Mechanizm pliku blokującego działa doskonale w warunkach, gdy jest zdefiniowane miejsce dla takich plików i wszystkie programy wiedzą, gdzie ich szukać. Niestety nie zawsze tak było w Linuksie. Korzystanie z tego mechanizmu stało się możliwe dopiero, gdy został zdefiniowany FSSTND (standard systemu plików Linuksa) z ustaloną lokalizacją plików blokujących, które zaczęły wtedy działać poprawnie dla urządzeń tty. Wcześniej zdarzyło się, że współistniało kilka możliwych lokalizacji plików blokujących wybranych przez programistów: */usr/spool/locks/*, */var/spool/locks/*, */var/lock/* i */usr/lock/*. Zamieszanie rodziło chaos. Programy otwierały pliki blokujące z różnych miejsc, a mające kontrolować jedno urządzenie tty. Efekt był taki, jak by pliki blokujące w ogóle nie były używane.

Aby rozwiązać ten problem, stworzono urządzenia *cua*. Zamiast polegać na plikach blokujących, które miały zabezpieczać przed kolidowaniem ze sobą programów korzystających z urządzeń szeregowych, zdecydowano, że to jądro będzie decydować, kto ma mieć dostęp do urządzenia. Jeżeli urządzenie *ttyS* było już otwarte, próba otwarcia *cua* kończyła się błędem. Program mógł to zinterpretować jako informację, że urządzenie jest używane. Jeżeli urządzenie *cua* było już otwarte i została podjęta próba otwarcia urządzenia *ttyS*, żądanie było blokowane, to znaczy wstrzymywane do czasu zamknięcia urządzenia *cua* przez inny proces. Działało do całkiem dobrze, jeżeli miałeś jeden modem skonfigurowany do odbierania połączeń i co jakiś czas chciałeś zadzwonić za pomocą tego samego urządzenia. Kłopoty pojawiły się w środowiskach, gdzie wiele programów chciało dzwonić z tego samego urządzenia. Jedynym sposobem na rozwiązanie tego problemu było zastosowanie plików blokujących. Powrót do punktu wyjścia.

Wystarczy wspomnieć, że przyszedł tu z pomocą standard systemu plików Linuksa. Teraz pliki blokujące muszą znajdować się w katalogu */var/lock* i nazywać zgodnie z przyjętą konwencją, czyli plik blokujący dla urządzenia *ttyS1* nazywa się na przykład *LCK..ttyS1*. Pliki blokujące *cua* powinny także znajdować się w tym katalogu, ale używanie urządzeń *cua* nie jest zalecane.

Przez jakiś czas urządzenia *cua* będą jeszcze funkcjonowały, by zapewnić kompatybilność w okresie przejściowym, ale stopniowo będą wycofywane. Jeżeli zastanawiasz się, czego używać, trzymaj się urządzeń *ttyS* i upewnij się, że twój system jest zgodny z FSSTND lub że przynajmniej wszystkie programy korzystające z urządzeń szeregowych umieszczają pliki blokujące w tym samym miejscu. Większość oprogramo-

wania pracującego z urządzeniami szeregowymi tty posiada opcję kompilacyjną pozwalającą na wskazanie miejsca umieszczania plików blokujących. Często występuje ona w postaci zmiennej o nazwie typu `LOCKDIR` w pliku *Makefile* lub w nagłówkowym pliku konfiguracyjnym. Jeżeli sam kompilujesz oprogramowanie, najlepiej jest ustawić tę zmienną tak, by zapewnić zgodność z lokalizacją określoną przez FSSTND. Jeżeli korzystasz ze skompilowanych plików binarnych i nie jesteś pewien, gdzie program zapisuje swoje pliki blokujące, możesz użyć poniższego polecenia, by uzyskać wskazówkę:

```
strings plikbinarny | grep lock
```

Jeżeli wskazana lokalizacja nie zgadza się z pozostałą częścią twojego systemu, staraj się utworzyć dowiązanie symboliczne z katalogu plików blokujących, którego chce używać dany program, do katalogu */var/lock*. Nie jest to zbyt eleganckie rozwiązanie, ale działa.

Pliki specjalne urządzenia szeregowego

Numery podrzędne są identyczne dla obu typów urządzeń szeregowych. Gdybyś miał swój modem na jednym z czterech standardowych portów COM, jego numer podrzędny byłby numerem portu COM plus 63. Gdybyś używał specjalnego urządzenia szeregowego, takiego jak szybki wieloportowy kontroler szeregowy, prawdopodobnie musiałbyś tworzyć dla niego specjalne pliki urządzeń. Zapewne karta taka nie posługiwałaby się standardowym sterownikiem urządzenia. Odpowiednie szczegóły zapewne znajdziesz w dokumencie *Serial-HOWTO*.

Załóżmy, że twój modem jest podłączony do COM2:. Jego numer podrzędny to 65, a nadrzędny to 4 w przypadku normalnego zastosowania. Powinno istnieć urządzenie *ttyS1*, które ma takie numery. Wylistuj urządzenia szeregowe tty w katalogu */dev/*. Piąta i szósta kolumna pokazują odpowiednio numery podrzędne i nadrzędne:

```
$ ls -l /dev/ttyS*
0 crw-rw---- 1 uucp dialout 4, 64 Oct 13   1997 /dev/ttyS0
0 crw-rw---- 1 uucp dialout 4, 65 Jan 26  21:55 /dev/ttyS1
0 crw-rw---- 1 uucp dialout 4, 66 Oct 13   1997 /dev/ttyS2
0 crw-rw---- 1 uucp dialout 4, 67 Oct 13   1997 /dev/ttyS3
```

Gdyby nie było urządzenia o numerze nadrzędnym 4 i podrzędnym 65, musiałbyś je stworzyć. W takiej sytuacji zaloguj się jako użytkownik uprzywilejowany i napisz:

```
# mknod -m 666 /dev/ttyS1 c 4 65
# chown uucp.dialout /dev/ttyS1
```

Dystrybucje Linuksa używają różnych strategii do określania, kto powinien być właścicielem urządzeń szeregowych. Czasem będą one własnością użytkownika *root*, a innym razem będą należały na przykład do **uucp**, tak jak w naszym przykładzie. Współczesne dystrybucje mają specjalną grupę dla urządzeń służących do dzwonienia. Każdy użytkownik, który ma prawo ich używać, jest dodawany do tej grupy.

Niektórzy sugerują stworzenie dowiązania symbolicznego */dev/modem* do urządzenia modemu, tak by zwykli użytkownicy nie musieli zapamiętywać czegoś tak skomplikowanego jak *ttyS1*. Jednak nie możesz używać w jednym programie nazwy *modem*, a w drugim rzeczywistej nazwy pliku urządzenia. Ich pliki blokujące będą miały różne nazwy i mechanizm blokowania nie zadziała.

Urządzenia szeregowe

RS-232 jest obecnie najbardziej znanym standardem komunikacji szeregowej w świecie PC. Wykorzystuje wiele układów do transmisji pojedynczych bitów oraz do synchronizacji. Można wprowadzić dodatkowe linie do sygnalizacji obecności nośnej (używanej przez modemy) i do uzgadniania (ang. *handshaking*). Linux obsługuje wiele kart szeregowych zgodnych ze standardem RS-232.

Uzgadnianie sprzętowe jest opcjonalne, ale bardzo przydatne. Pozwala obu stronom na sygnalizowanie gotowości odbioru kolejnych danych lub na powiadomienie, że druga strona powinna poczekać, aż odbiorca zakończy przetwarzanie odebranych danych. Linie używane do tego celu są nazywane odpowiednio „Clear to Send" (CTS) i „Ready to Send" (RTS), co wyjaśnia potoczną nazwę uzgadniania sprzętowego: RTS/CTS. Innym rodzajem uzgadniania, z którym mogłeś się już spotkać, jest XON/XOFF. Wykorzystuje ono dwa wyznaczone znaki, zwykle [CTRL+S] i [CTRL+Q] do sygnalizowania drugiej stronie, że powinna odpowiednio zatrzymać lub rozpocząć przesyłanie danych. Choć sposób ten jest łatwy do zaimplementowania i działa poprawnie na terminalach uproszczonych (ang. *dumb terminals*), powoduje zamieszanie w przypadku danych binarnych. Może się bowiem zdarzyć, że wolisz przesłać te znaki jako część strumienia danych i chcesz, aby były interpretowane jako znaki sterujące. Poza tym metoda ta jest wolniejsza niż uzgadnianie sprzętowe, które jako proste i szybkie jest zalecane zamiast XON/XOFF, o ile oczywiście masz wybór.

W pierwszych modelach IBM PC interfejs RS-232 był sterowany przez układ scalony UART 8250. PC z czasów procesora 486 używały nowszej wersji układu UART 16450. Był on nieco szybszy niż 8250. Prawie wszystkie komputery oparte na Pentium są wyposażone w jeszcze nowszą wersję układu UART 16550. Niektóre marki (przeważnie modemy wewnętrzne wyposażone w zestaw układów Rockwell) wykorzystują zupełnie inne układy emulujące zachowanie 16550 i mogą być traktowane podobnie. Standardowy sterownik portu szeregowego Linuksa obsługuje je wszystkie*.

Układ 16550 jest znacznym krokiem naprzód w stosunku do 8250 i 16450, ponieważ oferuje 16-bajtowy bufor FIFO. 16550 jest w rzeczywistości rodziną urządzeń UART, do której należą układy 16550, 16550A i 16550AFN (nazwa została później zmie-

* Zauważ, że nie mówimy tu o tak zwanych WinModemach! WinModemy mają bardzo prostą budowę sprzętową i do wykonania całej pracy w pełni wykorzystują główny procesor, zamiast dedykowanych układów. Zdecydowanie *odradzamy* ci zakup takiego modemu – kup prawdziwy modem. Linux oczywiście obsługuje WinModemy, ale nie jest to atrakcyjne rozwiązanie.

niona na PCI16550DN). Różn ice mię dzy nimi po leg ają na za pewn ieniu działania FI-FO; w układzie 16550AFN działa ono na pew no. Ist niał ta kże układ NS16550, ale w nim bu for FIFO nig dy tak na prawdę nie działał.

Układy UART 8250 i 16450 miały pro sty bu for jed nob ajto wy. Oznac zało to, że 16450 generował prze rwan ie dla ka żd ego na dan ego lub odeb rane go zna ku. Ka żde wy-magało kr ótk iego cza su na jego obsługę i to nie wielk ie opó źnien ie ogran iczało pręd-kość układu 16450 do 9600 bitów na se kundę w ty pow ym kom put erze z ma gis tralą ISA.

W do myś lnej kon fig ura cji jądro spraw dza czte ry stan dard owe por ty sze reg owe, od COM1: do COM4:. Jądro jest ta kże w sta nie wy kryć, jaki układ UART jest używ any dla każdego ze standardowych portów szeregowych i wykorzystuje bufor FIFO układu 16550, je żeli jest do stępny.

Używanie narzędzi konfiguracyjnych

Te raz przyj rzyj my się krótko dwóm naj bar dziej przy dat nym na rzę dziom do kon fi-gu ra cji urządze nia sze re go we go: *setserial* i *stty*.

Poleceniesetserial

Jądro zro bi wszyst ko co w je go mo cy, by po praw nie roz po znać kon fi gu ra cję two je-go urządze nia sze re go we go, ale wie lość mo żli wo ści po wo du je, że trud no jest uzy-skać w prak ty ce stu pro cen tową nie za wod ność. Do brym przykładem te go, co spra-wia pro ble my, są mo de my we w nętrz ne, o kt ó rych mó w iliśmy wcze śniej. Uży wa-ny przez nie układ UART ma 16-bajtowy bufor FIFO, ale z punktu widzenia sterow ni ka urządze nia w jądrze wygląda jak układ UART 16450: dopóki nie wska-żemy ste row ni ko wi, że jest to urządze nie 165 50, jądro nie bę dzie wy ko rzy sty wać rozszerzonego bufora. Innym przykładem są uproszczone karty 4-portowe po-zwalające na współdzielenie jednego IRQ przez wiele urządzeń szeregowych. W ta kiej sy tu acji mu si my wska zać jądru właści we IRQ i uprze dzić je, że IRQ może być ws pół dzielone.

Do kon fi gu ra cji ste row ni ka sze re go we go w cza sie pra cy stwo rzo no pro gram *set se-rial*. Po le ce nie to jest po wszech nie uru cha mia ne w cza sie star tu sys te mu ze skryp tu *0setserial* lub *rc.serial*, w za le żno ści od dys try bu cji. Skrypt ma tak usta wić ini cja cję sterownika szeregowego, aby ten dostosował się do niestandardowych lub nie-zwykłych urządzeń sze re go wych za insta lo wa nych w kom pu te rze.

Ogó lna skład nia po lec enia *setserial* jest na stęp ująca:

```
setserial urz□dzenie [parametry]
```

gdzie *urządzenie* to jed no z urządzeń sze reg owy ch, na przykład *ttyS0*.

Po lec enie *setserial* ma wie le par ametrów. Naj pop ula rnie jsze z nich opis ano w ta beli 4-1. In form acje o po zos tałych znaj dziesz w podręcz niku elek tro nicznym *setserial*.

Tabela 4-1. Parametry polecenia setserial

Parametr	*Opis*
`port numer_portu`	Określa adres portu wejścia/wyjścia urządzenia szeregowego. Numery portu powinny być podawane w notacji szesnastkowej, tzn. `0x2f8`.
`irq` *`numer`*	Określa numer przerwania używany przez urządzenie szeregowe.
`uart` *`typ_uart`*	Określa typ UART urządzenia szeregowego. Powszechnie stosowane wartości to `16450`, `16550` itd. Ustawienie tej wartości na `none` wyłącza dane urządzenie szeregowe.
`fourport`	Określenie tego parametru mówi sterownikowi szeregowemu jądra, że port jest jednym z portów czteroportowej karty AST.
`spd_hi`	Programuje UART na prędkość 57,6 kb/s, gdy proces żąda 38,4 kb/s.
`spd_vhi`	Programuje UART na prędkość 115 kb/s, gdy proces żąda 38,4 kb/s.
`spd_normal`	Programuje UART na domyślną prędkość 38,4 kb/s, gdy zostanie zażądana. Parametr ten jest używany do wyłączenia działania `spd_hi` i `spd_vhi` wykonanych na danym urządzeniu szeregowym.
`auto_irq`	Parametr ten powoduje, że jądro próbuje automatycznie określić IRQ danego urządzenia. Próba może się nie powieść, a więc lepiej traktować to jako żądanie odgadnięcia IRQ przez jądro. Jeżeli znasz IRQ urządzenia, powinieneś od razu użyć opcji `irq`.
`autoconfig`	Parametr ten musi być określony w połączeniu z parametrem `port`. Podanie tego parametru powoduje, że *setserial* zleca jądru próbę automatycznego określenia typu układu UART znajdującego się pod zadanym adresem portu. Jeżeli zostanie podany również parametr `auto_irq`, jądro podejmie także próbę automatycznego wykrycia IRQ.
`skip_test`	Parametr ten mówi jądru, aby nie wykonywało sprawdzania typu układu UART podczas automatycznej konfiguracji. Jest on niezbędny, jeżeli układ UART nie jest poprawnie wykrywany przez jądro.

Plik *rc* konfigurujący porty szeregowe w czasie uruchamiania komputera może wyglądać tak, jak w przykładzie 4-1. W większości dystrybucji Linuksa będzie on bardziej wyrafinowany niż tutaj.

Przykład 4-1. Przykładowy plik rc.serial zawierający polecenia setserial

```
# /etc/rc.serial - skrypt konfiguruj□cy □□cze szeregowe
#
# Konfiguracja urz□dze□ szeregowych
/sbin/setserial /dev/ttyS0 auto_irq skip_test autoconfig
/sbin/setserial /dev/ttyS1 auto_irq skip_test autoconfig
/sbin/setserial /dev/ttyS2 auto_irq skip_test autoconfig
/sbin/setserial /dev/ttyS3 auto_irq skip_test autoconfig
#
# Wy□wietlenie konfiguracji urz□dze□ szeregowych
/sbin/setserial -bg /dev/ttyS*
```

Argument `-bg /dev/ttyS*` w ostatnim poleceniu wypisze ładnie sformatowane podsumowanie konfiguracji wszystkich urządzeń szeregowych. Wynik będzie wyglądał tak, jak w przykładzie 4-2.

Przykład 4-2. Wynik polecenia setserial -bg /dev/ttyS

```
/dev/ttyS0 at 0x03f8 (irq = 4) is a 16550A
/dev/ttyS1 at 0x02f8 (irq = 3) is a 16550A
```

Polecenie stty

Nazwa *stty* może oznaczać „set tty", ale polecenie *stty* bywa też używane do wyświetlania konfiguracji terminala. Polecenie *stty*, prawdopodobnie jeszcze bardziej niż *setserial*, wprawia w konsternację posiadaną liczbą charakterystyk, które można konfigurować. W tej chwili pokażemy najważniejsze z nich. Pozostałe znajdziesz na stronie podręcznika elektronicznego *stty*.

Polecenie *stty* jest najczęściej używane do konfigurowania parametrów terminala, które decydują na przykład, czy wprowadzane znaki będą wyświetlane na ekranie albo czy klawisz powinien generować sygnał przerwania. Wcześniej wyjaśniliśmy, że urządzenia szeregowe są urządzeniami tty i dlatego polecenie *stty* odnosi się także do nich.

Jednym z najważniejszych zastosowań *stty* w urządzeniach szeregowych jest włączenie uzgadniania sprzętowego w urządzeniu. Wcześniej krótko wspomnieliśmy o uzgadnianiu sprzętowym. Domyślna konfiguracja urządzeń szeregowych zakłada wyłączenie uzgadniania sprzętowego. Wówczas mogą działać kable szeregowe „trzyżyłowe". Nie obsługują one sygnałów wymaganych do uzgadniania sprzętowego i gdyby było ono domyślnie włączone, nie można byłoby przez nie przesłać żadnych znaków, by to zmienić.

Co dziwniejsze, niektóre szeregowe programy komunikacyjne nie włączają uzgadniania sprzętowego, a więc jeżeli twój modem je obsługuje, powinieneś go skonfigurować tak, żeby go używał (od szukaj w instrukcji modemu właściwe polecenie), a także skonfiguruj odpowiednio urządzenie szeregowe. Polecenie *stty* ma znacznik `crtscts`, który włącza uzgadnianie sprzętowe w urządzeniu – będziesz musiał go użyć. Polecenie prawdopodobnie najlepiej uruchomić z pliku *rc.serial* (lub równoważnego) w czasie startu systemu za pomocą poleceń pokazanych w przykładzie 4-3.

Przykład 4-3. Przykładowe polecenia stty w pliku rc.serial

```
#
stty crtscts < /dev/ttyS0
stty crtscts < /dev/ttyS1
stty crtscts < /dev/ttyS2
stty crtscts < /dev/ttyS3
#
```

Polecenie *stty* działa domyślnie na bieżącym terminalu, ale używając funkcji przekierowującej wejście („<") powłoki, możemy za pomocą *stty* operować na dowolnym urządzeniu tty. Znak przekierowania „<" często bywał mylony z „>" – na szczęście nowsze wersje *stty* mają dużo prostszą składnię takiego przekierowania.

Aby użyć nowej składni, musimy napisać naszą przykładową konfigurację tak, jak w przykładzie 4-4.

Przykład 4-4. Przykład polecenia stty w pliku rc.serial z wykorzystaniem nowej składni

```
#
stty crtscts -F /dev/ttys0
stty crtscts -F /dev/ttys1
stty crtscts -F /dev/ttys2
stty crtscts -F /dev/ttys3
#
```

Wspomnieliśmy, że polecenie *stty* może być używane do wyświetlenia parametrów konfiguracyjnych terminala. Aby wyświetlić wszystkie aktywne ustawienia urządzenia tty, użyj:

```
$ stty -a -F /dev/ttyS1
```

Wynik działania tego polecenia, przedstawiony jako przykład 4-5, pokazuje stan wszystkich znaczników urządzenia. Znacznik poprzedzony znakiem minus, na przykład *-crtscts*, oznacza, że dana opcja jest wyłączona.

Przykład 4-5. Wynik działania polecenia stty -a

```
speed 19200 baud; rows 0; columns 0; line = 0;
intr = ^C; quit = "\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
         eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R;
         werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 hupcl -cstopb cread clocl -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr -icrnl -ixon
        -ixoff -iuclc -ixany -imaxbel
-opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0
         bs0 vt0 ff0
-isig -icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop
         -echoprt echoctl echoke
```

Opis najważniejszych znaczników znajduje się w tabeli 4-2. Każdy z nich jest włączany przez podanie w poleceniu *stty* i wyłączany przez podanie w poleceniu *stty* z poprzedzającym znakiem -. Zatem, aby wyłączyć uzgadnianie sprzętowe na urządzeniu `ttyS0`, napisałbyś:

```
$ stty -crtscts -F /dev/ttyS0
```

Kolejny przykład łączy niektóre z tych znaczników i konfiguruje urządzenie *ttyS0* na 19200 bitów na sekundę, 8 bitów danych, brak parzystości i uzgadnianie sprzętowe bez wypisywania odebranych znaków u nadawcy:

```
$ stty 19200 cs8 -parenb -crtscts -echo -F /dev/ttyS0
```

Tabela 4-2. Najważniejsze znaczniki w konfiguracji urządzeń szeregowych

Znaczniki	*Opis*
`N`	Ustawienie prędkości łącza na `N` b/s.
`crtscts`	Włączenie/wyłączenie uzgadniania sprzętowego.
`ixon`	Włączenie/wyłączenie kontroli przepływu XON/XOFF.

Znaczniki	*Opis*
`clocal`	Włączenie/wyłączenie sygnałów sterowania modemem, takich jak DTR/DTS i DVD. Jest to potrzebne, jeżeli używasz „trzy żyłowego” kabla szeregowego.
`cs5 cs6 cs7 cs8`	Ustawienie liczby bitów danych odpowiednio na 5, 6, 7 lub 8.
`parodd`	Włączenie dopełniania bajtu do nieparzystej liczby jedynek. Wyłączenie tego znacznika powoduje włączenie dopełniania bajtu do parzystej liczby jedynek.
`parenb`	Włączenie sprawdzania parzystości. Zanegowanie tego znacznika powoduje nieużywanie parzystości.
`cstopb`	Włączenie używania dwóch bitów stopu na znak. Zanegowanie tego znacznika powoduje używanie jednego bitu stopu na znak.
`echo`	Włączenie/wyłączenie wysyłania odebranych znaków do nadawcy.

Urządzenia szeregowe i monit login:

Swego czasu instalacja Uniksa wymagała najczęściej jednego serwera i wielu „uproszczonych” terminali znakowych lub modemów do połączeń komutowanych. Obecnie ten typ instalacji jest mniej powszechny. To dobra wiadomość dla wielu osób zainteresowanych taką metodą pracy, ponieważ „uproszczone” terminale można teraz kupić za grosze. Konfiguracje z modemami nie straciły na popularności, ale dzisiaj są one raczej używane do obsługi logowania przez SLIP lub PPP (co omawiamy w rozdziale 7, *IP łącza szeregowego*, i w rozdziale 8, *Protokół punkt-punkt*), a nie do zwykłego logowania. Niemniej jednak każda z tych konfiguracji może wykorzystywać prosty program o nazwie *getty*.

Określenie *getty* można traktować jako skrót od „get tty”. Program *getty* otwiera urządzenie szeregowe, konfiguruje je w odpowiedni sposób, opcjonalnie konfiguruje modem i czeka na zestawienie połączenia. Aktywne połączenie na urządzeniu szeregowym jest zwykle wskazywane przez wyprowadzenie DCD (*Data Carrier Detect*) wzbudzanego urządzenia szeregowego.

Gdy zostanie wykryte połączenie, program *getty* wyświetla monit `login:` i wywołuje program *login*, aby obsłużył rzeczywiste logowanie do systemu. Każdy terminal wirtualny (na przykład *dev/tty1*) w Linuksie posiada działający na jego rzecz program *getty*.

Istnieje szereg różnych implementacji *getty*, a każda z nich jest zaprojektowana tak, aby pewne konfiguracje obsługiwać lepiej niż inne. Opisywany tutaj *getty* nosi nazwę *mgetty*. Jest dosyć popularny, ponieważ posiada wszelkie funkcje, które czynią go szczególnie przydatnym do obsługi modemów. Między innymi jest wyposażony w programy do automatycznej obsługi faksu i modemów głosowych. Skoncentrujemy się na konfigurowaniu *mgetty* do odpowiadania na typowe połączenia mające na celu transmisję danych, a całą resztę pozostawimy ci do zbadania w wolnej chwili.

Konfigurowanie demona mgetty

Demon *mgetty* jest dostępny w postaci źródłowej pod adresem *ftp://alpha.greenie.net/pub/mgetty/source/*, a w każdej dystrybucji Linuksa występuje w postaci pakietu. Demon *mgetty* różni się od większości pozostałych implementacji *getty* tym, że został stworzony specjalnie dla modemów kompatybilnych ze standardem Hayesa. Wciąż obsługuje bezpośrednie połączenia terminalowe, ale najlepiej nadaje się do aplikacji pracujących po łączu komutowanym. Zamiast używać linii DCD do wykrywania przychodzącego połączenia, oczekuje na komunikat `RING` generowany w momencie wykrycia nadchodzącego połączenia przez nowoczesne modemy, o ile nie są skonfigurowane na automatyczne odpowiadanie.

Główny program wykonywalny nazywa się */usr/sbin/mgetty*, a jego plik konfiguracyjny to */etc/mgetty/mgetty.config*. Poza tym jest szereg innych programów binarnych i plików konfiguracyjnych, które obsługują inne funkcje *mgetty*.

W większości instalacji konfiguracja polega na edycji pliku */etc/mgetty/mgetty.config* i dodaniu odpowiednich wpisów w pliku */etc/inittab*, automatycznie uruchamiających *mgetty*.

Przykład 4-6 pokazuje bardzo prosty plik konfiguracyjny *mgetty* dla dwóch urządzeń szeregowych. Pierwsze urządzenie */dev/ttyS0*, obsługuje modem kompatybilny ze standardem Hayesa przy prędkości 38 400 bps. Drugie, */dev/ttyS1*, obsługuje bezpośrednio podłączony terminal VT100 z prędkością 19200 bps.

Przykład 4-6. Przykładowy plik /etc/mgetty/mgetty.config

```
#
# plik konfiguracyjny mgetty
#
# jest to przyk□adowy plik konfiguracyjny, wi□cej szczegó□ów
# znajdziesz w mgetty.info
#
# wiersze komentarza zaczynaj□ si□ od "#", puste wiersze s□
# ignorowane
#
# ----- sekcja ogólna -----
#
# w tej sekcji umieszczasz ogólne warto□ci domy□lne, dane
# dotycz□ce portu znajduj□ si□ dalej
#
# modem pracuje z pr□dko□ci□ 38400 bps
speed 38400
#
# ustawienie ogólnego poziomu debugowania na "4" (domy□lnie z
# policy.h)
debug 4
#

# ----- sekcja okre□laj□ca porty -----
#
# Tutaj mo□esz umie□ci□ ustawienia dotycz□ce tylko danej linii
# i nie dotycz□ce innych
#
#
# Modem Hayesa pod□□czony do ttyS0: bez obs□ugi faksu, bez
```

```
# logowania
#
port ttyS0
  debug 3
  data-only y
#
# bezpo□rednie pod□□czenie terminala VT100, który potrzebuje
# DTR
#
port ttyS1
  direct y
  speed 19200
  toggle-dtr n
#
```

Plik konfiguracyjny zawiera opcje ogólne i specyficzne dla portów. W naszym przykładzie użyliśmy opcji ogólnych do ustawienia prędkości na 38 400 bitów na sekundę. Wartość ta jest dziedziczona przez port *ttyS0*. To ustawienie prędkości dotyczy portów *mgetty*, dopóki nie zostanie ono nadpisane przez ustawienie prędkości specyficznej dla portu, co robimy w konfiguracji *ttyS1*.

Słowo kluczowe `debug` kontroluje liczbę informacji logowanych przez *mgetty*. Słowo kluczowe `data-only` w konfiguracji *ttyS0* powoduje, że *mgetty* ignoruje wszelkie funkcje faksowe modemu, i w związku z tym modem przesyła tylko dane. Słowo kluczowe `direct` w konfiguracji *ttyS1* mówi programowi *mgetty*, aby nie próbował inicjować modemu na danym porcie. Wreszcie słowo kluczowe `toggle-dtr` mówi *mgetty*, aby nie próbował zawieszać linii przy braku sygnału DTR (*Data Terminal Ready*) na interfejsie szeregowym. Niektóre terminale tego nie lubią.

Możesz także zdecydować się na pozostawienie pustego pliku *mgetty.config*, a większość z tych parametrów określić za pomocą argumentów wiersza poleceń. Dokumentacja dotycząca aplikacji zawiera pełny opis parametrów pliku konfiguracyjnego *mgetty* i argumentów wiersza poleceń. (Patrz fragment pliku poniżej).

Aby uaktywnić tę konfigurację, musimy dodać dwa wpisy do pliku */etc/inittab*. Plik *inittab* jest plikiem konfiguracyjnym polecenia *init* Uniksa w wersji System V. Polecenie *init* jest odpowiedzialne za inicjację systemu. Zapewnia automatyczne uruchamianie programów w czasie startu systemu i ponowne ich uruchamianie po zakończeniu pracy. Rozwiązanie to idealnie nadaje się do obsługi programu *getty*.

```
T0:23:respawn:/sbin/mgetty ttyS0
T1:23:respawn:/sbin/mgetty ttyS1
```

Każdy wiersz pliku */etc/inittab* zawiera cztery pola oddzielone dwukropkami. Pierwsze pole to identyfikator jednoznacznie określający wpis w pliku. Tradycyjnie jest on dwuznakowy, ale nowsze wersje pozwalają na zastosowanie czterech znaków. Drugie pole to lista poziomów uruchomienia (ang. *run levels*), przy których wpis powinien być aktywny. Poziom uruchomienia pozwala na różne konfiguracje systemu i jest zaimplementowany za pomocą drzewiastej struktury skryptów startowych, znajdujących się w katalogach */etc/rc1.d*, */etc/rc2.d* itd. Funkcja ta jest zwykle implementowana w bardzo prosty sposób i powinieneś wzorować swoje wpisy na innych wpisach w pliku lub zajrzeć do dokumentacji, jeżeli potrzebujesz dodatkowych in-

formacji. Trzecie pole opisuje, kiedy zadziałać. W przypadku uruchamiania programu *getty*, pole to powinno mieć wartość `respawn`, co oznacza, że polecenie powinno być automatycznie uruchomione ponownie po zakończeniu działania. Istnieje kilka innych możliwości, ale nie są dla nas teraz przydatne. Czwarte pole to rzeczywiste polecenie do wykonania. Tutaj wpisujemy polecenie *mgetty* i jego argumenty. W naszym prostym przykładzie inicjujemy i ponownie uruchamiamy *mgetty*, gdy system działa na poziomie drugim lub trzecim, a jako argumenty podajemy nazwę urządzenia, z którego chcemy korzystać. Polecenie *mgetty* automatycznie zakłada ścieżkę */dev/*, a więc nie musimy jej tu taj podawać.

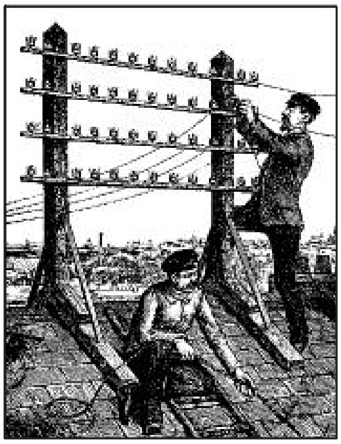
Podrozdział ten był krótkim omówieniem *mgetty* i sposobu udostępniania monitu logowania dla urządzeń szeregowych. Więcej na ten temat możesz znaleźć w dokumencie *Serial-HOWTO*.

Gdy dokonasz edycji plików konfiguracyjnych, musisz przeładować proces *init*, aby zmiany były aktywne. Po prostu wyślij sygnał hangup do procesu *init*. Proces ten zawsze ma ID równe 1, a więc możesz bezpiecznie wydać następujące polecenie:

```
# kill -HUP 1
```

5

Konfigurowanie sieci TCP/IP



W tym rozdziale pokażemy wszystkie kroki niezbędne do skonfigurowania sieci TCP/IP na twoim komputerze. Zaczniemy od przypisania adresów IP, potem zajmiemy się konfigurowaniem interfejsów sieciowych TCP/IP i na koniec przedstawimy kilka narzędzi, które przydają się przy rozwiązywaniu problemów z siecią.

Większość zadań omówionych w tym rozdziale wykonuje się zwykle tylko raz. Po pliki konfiguracyjne sięga się powtórnie tylko wtedy, gdy dodaje się nowy system do sieci lub zupełnie przekonfigurowuje istniejący system. Niektóre polecenia używane do konfigurowania TCP/IP muszą być jednak uruchamiane przy każdym starcie systemu, zwykle przez ich wywołanie ze skryptów */etc/rc**.

Część sieciowa procedury konfiguracyjnej zwykle jest zawarta w skrypcie. Jego nazwa zależy od dystrybucji Linuksa. W wielu starszych dystrybucjach był to skrypt *rc.net* lub *rc.inet*. Czasem spotkasz się także z dwoma skryptami o nazwach *rc.inet1* i *rc.inet2* – pierwszy z nich inicjuje część sieciową związaną z jądrem, a drugi uruchamia podstawowe usługi sieciowe i aplikacje. We współczesnych dystrybucjach pliki *rc* są uporządkowane w bardziej wyrafinowany sposób. W katalogu */etc/init.d/* (lub */etc/rc.d/rc.init.d/*) możesz znaleźć skrypty tworzące urządzenia sieciowe, a inne pliki *rc* mogą uruchamiać aplikacje sieciowe. Przykłady w tej książce zostały oparte właśnie na tym ostatnim porządku plików.

Niniejszy rozdział przedstawia części skryptu konfigurujące interfejsy sieciowe, natomiast aplikacje zostaną omówione w dalszych rozdziałach. Po lekturze tego rozdziału będziesz znać zestaw poleceń, za pomocą których poprawnie skonfigurujesz sieć TCP/IP na swoim komputerze. Zatem powinieneś zastąpić wszelkie przykładowe polecenia w swoich skryptach konfiguracyjnych własnymi, upewnić się, że skrypt jest uruchamiany z podstawowego skryptu *rc* w czasie uruchamiania systemu i ponownie uruchomić komputer. Skrypty sieciowe *rc* zawarte w twojej ulubionej dystrybucji Linuksa powinny zawierać idealny przykład, z którego możesz skorzystać.

Montowanie systemu plików /proc

Niektóre narzędzia konfiguracyjne NET-2 i NET-3 w Linuksie komunikują się z jądrem za pomocą systemu plików */proc*. Interfejs ten pozwala na dostęp do roboczych informacji jądra przez mechanizm przypominający system plików. Po jego zamontowaniu możesz oglądać pliki tak jak w każdym innym systemie plików lub wyświetlać ich zawartość. Do typowych elementów należą: plik *loadavg* informujący o średnim obciążeniu systemu i plik *meminfo* pokazujący aktualne wykorzystanie pamięci głównej i pamięci wymiany.

Do tego wszystkiego kod sieciowy dodaje katalog *net*. Zawiera on szereg plików pokazujących takie rzeczy, jak tablica ARP jądra, stan połączeń TCP oraz tablice rutingu. Większość narzędzi do administrowania siecią odczytuje potrzebne im informacje właśnie z tych plików.

System plików *proc* (znany też pod nazwą *procfs*) zwykle jest montowany w katalogu */proc* w czasie uruchamiania systemu. Najlepszym sposobem na zrobienie tego jest dodanie następującego wiersza w pliku */etc/fstab*:

```
# punkt montowania procfs
none      /proc      proc      defaults
```

a następnie wykonanie *mount /proc* ze skryptu */etc/rc*.

Obecnie *procfs* jest skonfigurowany domyślnie w większości jąder. Jeżeli w twoim jądrze nie ma *procfs*, otrzymasz komunikat typu: `mount: fs type procfs not supported by kernel`. Będziesz zatem musiał przekompilować jądro i odpowiedzieć „yes” na pytanie o obsługę *procfs*.

Instalowanie plików binarnych

Jeżeli korzystasz z dowolnej gotowej dystrybucji Linuksa, znajdziesz w niej główne aplikacje i narzędzia sieciowe wraz z odpowiednim zestawem przykładowych plików. Jedynym przypadkiem, w którym może zajść potrzeba znalezienia i zainstalowania nowych narzędzi, jest instalacja nowej wersji jądra. Nowe jądro miewa zmiany w warstwie sieciowej i dlatego należy uaktualnić podstawowe narzędzia konfiguracyjne. Uaktualnienie takie oznacza przynajmniej ponowną kompilację, ale czasem także wymaga nowego zestawu plików binarnych. Są one dostępne na oficjalnej witrynie macierzystej *ftp.inka.de/pub/comp/Linux/networking/NetTools/* w postaci pakietu *net-tools-XXX.tar.gz*, gdzie *XXX* to numer wersji. Wersja dla Linuksa 2.0 nosi nazwę *net-tools-1.45*.

Gdybyś chciał skompilować i zainstalować samodzielnie standardowe aplikacje sieciowe TCP/IP, mógłbyś zdobyć ich źródła z większości serwerów FTP Linuksa. Wszystkie współczesne dystrybucje Linuksa zawierają pełny zestaw aplikacji sieciowych TCP/IP, takich jak przeglądarka WWW, programy *telnet* i *ftp* oraz inne aplikacje sieciowe, na przykład *talk*. Jeżeli jednak dojdziesz do wniosku, że coś musisz skompilować samodzielnie, prawdopodobnie nie będziesz miał z tym problemów w Linuksie, jeżeli tylko będziesz postępował zgodnie z instrukcjami zawartymi w pakiecie źródłowym.

Ustalanie nazwy hosta

Większość aplikacji sieciowych, jeżeli nie wszystkie, oczekuje od ciebie ustawienia lokalnej nazwy hosta na jakąś sensowną wartość. Najlepiej zrobić to w czasie procedury uruchamiania systemu przez wykonanie polecenia *hostname*. Aby ustalić nazwę hosta `nazwa`, wprowadź:

```
# hostname nazwa
```

Powszechnie stosuje się skróconą nazwę hosta bez podawania nazwy domeny. Na przykład hosty w wirtualnym browarze (opisanym w dodatku A, *Przykładowa sieć: browar wirtualny*) mogłyby nosić nazwy **vale.vbrew.com** czy **vlager.vbrew.com**. Są to ich *pełne nazwy domenowe* (ang. *fully qualified domain names* – FQDN). Nazwa hosta to jedynie pierwszy człon pełnej nazwy, czyli na przykład **vale**. Jednak choć lokalna nazwa hosta jest często używana do szukania jego adresu IP, musisz mieć pewność, że biblioteka resolvera będzie w stanie znaleźć adres IP hosta. Zwykle oznacza to, że musisz wprowadzić nazwę do pliku */etc/hosts*.

Niektórzy zalecają użycie polecenia *domainname*, by powiadomić jądro o nazwie domeny, czyli o pozostałej części FQDN. Wydaje się, że można by połączyć wynik *hostname* i *domainname*, aby uzyskać pełną nazwę domenową. Jednak w najlepszym razie rezultat byłby tylko w połowie poprawny. Polecenie *domainname* jest bowiem używane do definiowania domeny NIS, która może być zupełnie inna niż domena DNS, do której należy host. Dlatego go warto zadbać o to, aby nazwa hosta była rozwiązywalna przez wszystkie nowe wersje polecenia *hostname*. W tym celu należy dodać wpis do lokalnego serwera nazw domen lub umieścić pełną nazwę domenową w pliku */etc/hosts*. Teraz możesz użyć argumentu `–fqdn` polecenia *hostname*, aby wyświetlić pełną nazwę domenową.

Przypisywanie adresu IP

Jeżeli nie planujesz pracy w sieci, ale konfigurujesz oprogramowanie sieciowe na swoim hoście, aby na przykład móc uruchomić oprogramowanie INN Netnews, możesz bezpiecznie pominąć ten podrozdział, ponieważ jedynym potrzebnym ci adresem IP będzie interfejs pętli zwrotnej, który zawsze ma numer **127.0.0.1**.

Rzeczy nieco bardziej się komplikują w rzeczywistych sieciach takich jak Ethernet. Gdybyś chciał podłączyć swój host do istniejącej sieci, musiałbyś poprosić administratorów o nadanie ci w niej adresu IP. Jeżeli konfigurujesz sam całą sieć, musisz sam przypisać adresy IP.

Hosty w sieci lokalnej zwykle powinny mieć adresy z tej samej logicznej sieci IP. W związku z tym musisz przypisać adres IP dla sieci. Jeżeli masz kilka sieci fizycznych, musisz przypisać im różne numery sieci albo użyć podsieci i podzielić posiadany zakres adresów IP na kilka podsieci. Podsieci zostaną omówione w najbliższym podrozdziale.

Wybór numeru IP sieci w dużym stopniu zależy od tego, czy w niedalekiej przyszłości zamierzasz podłączyć się do Internetu. Jeżeli tak, powinieneś uzyskać

oficjalny adres IP *już teraz*. Poproś o pomoc swojego dostawcę usług internetowych. Jeżeli chcesz uzyskać numer sieci po prostu na wypadek, gdybyś kiedyś podłączył się do Internetu, poproś o formularz prośby o adres sieci, pisząc na adres *hostmaster@internic.net* lub zgłoś się do centrum informacji sieciowej w twoim kraju, o ile takie istnieje.

Jeżeli twoja sieć nie jest podłączona do Internetu i nie nosisz się z takim zamiarem, możesz wybrać dowolny do puszczalny adres sieci. Wystarczy upewnić się, że żadne pakiety z twojej sieci wewnętrznej nie przedostaną się do prawdziwego Internetu. Aby zagwarantować, że nic się nie stanie, nawet jeżeli pakiety się przedostaną, powinieneś użyć jednego z numerów sieci zarezerwowanych dla sieci prywatnych. Organizacja zajmująca się przydzielaniem numerów w Internecie (*Internet Assigned Numbers Authority* – IANA) wyznaczyła kilka adresów sieci z klasy A, B i C, których możesz używać bez rejestrowania. Adresy te są ważne tylko w twojej sieci prywatnej i nie są rutowane pomiędzy prawdziwymi ośrodkami w Internecie. Są one zdefiniowane w RFC 1597. My zamieściliśmy je w tabeli 2-1 w rozdziale 2, *Wybrane problemy sieci TCP/IP*. Zauważ, że druga i trzecia klasa zawierają odpowiednio 16 i 256 sieci.

Wybranie swojego adresu w jednej z tych sieci sprawdza się nie tylko w przypadku sieci zupełnie nie podłączonych do Internetu. Będąc w takiej sieci, możesz zaimplementować nieco bardziej ograniczony dostęp za pomocą jednego hosta jako gatewaya. Dla twojej sieci lokalnej gateway jest dostępny pod swoim wewnętrznym adresem IP, natomiast dla świata zewnętrznego – pod oficjalnie zarejestrowanym adresem (nadanym ci przez dostawcę). Powrócimy do tego rozwiązania przy omawianiu funkcji maskowania (ang. *masquarading*) w rozdziale 11, *Maskowanie IP i translacja adresów sieciowych*.

Na potrzeby naszych rozważań zakładamy, że administrator przykładowej sieci browarów używa numeru sieci z klasy B, powiedzmy **172.16.0.0**. Oczywiście numer z klasy C w zupełności by wystarczył zarówno dla sieci browarów, jak i winiarni. Klasy B używamy tu dla uproszczenia. Dzięki temu przykłady podziału na podsieci pokazane w następnym podrozdziale będą bardziej przekonujące.

Tworzenie podsieci

Aby obsłużyć kilka sieci Ethernet (lub innych, dla których dostępny jest sterownik), musisz podzielić swoją sieć na podsieci. Zauważ, że podział taki jest potrzebny tylko wtedy, gdy masz więcej niż jedną *sieć rozgłoszeniową* – łącza punkt-punkt się nie liczą. Na przykład gdybyś miał jedną sieć Ethernet i przynajmniej jedno łącze SLIP do świata zewnętrznego, nie musiałbyś dzielić swojej sieci na podsieci. Wyjaśniamy to bardziej szczegółowo w rozdziale 7, *IP łącza szeregowego*.

Aby obsłużyć dwie sieci Ethernet, administrator sieci browaru zdecydował się przeznaczyć w adresie 8 bitów części hosta na dodatkowe bity podsieci. Dla hosta zostaje 8 bitów, co pozwala na umieszczenie w każdej podsieci po 254 hosty. Następnie sieć numer 1 została przypisana do browaru, a sieć numer 2 do winiarni. Odpowiednie adresy sieci to **172.16.1.0** i **172.16.2.0**. Maska podsieci to **255.255.255.0**.

Gateway pomiędzy tymi dwoma sieciami, **vlager**, ma w obu sieciach numer hosta 1, co daje adresy IP odpowiednio **172.16.1.1** i **172.16.2.1**.

W tym przykładzie używamy dla uproszczenia sieci klasy B, choć sieć klasy C byłaby bardziej realistyczna. W nowym kodzie sieciowym jądra podział na podsieci nie zależy od granic bajtowych, a więc nawet klasa C może być dzielona na kilka podsieci. Na przykład mógłbyś użyć dwóch bitów części hosta na adres sieci, co dałoby 4 możliwe podsieci, po 64 hosty w każdej*.

Tworzenie plików hosts i networks

Jeśli już podzieliłeś swoją sieć na podsieci, powinieneś postarać się o proste rozwiązywanie nazw hostów za pomocą pliku */etc/hosts*. Jeżeli nie zamierzasz korzystać z DNS-u lub NIS-a do rozwiązywania adresów, musisz umieścić wszystkie hosty w pliku *hosts*.

Nawet jeżeli będziesz używał DNS-u lub NIS-a w czasie normalnej pracy, w pliku */etc/hosts* powinieneś mieć pewien podzbiór wszystkich nazw hostów. Musisz zapewnić rozwiązywanie nazw, nawet jeżeli nie działają żadne interfejsy sieciowe – na przykład w czasie uruchamiania systemu. Chodzi nie tylko o wygodę, ale też o możliwość zastosowania symbolicznych nazw hostów w skryptach sieciowych *rc*. Dzięki temu gdy zmienisz adresy IP, wystarczy jedynie skopiować uaktualniony plik *hosts* na wszystkie komputery i uruchomić je ponownie, zamiast edytować mnóstwo plików *rc*. Zwykle w pliku *hosts* umieszczasz wszystkie lokalne nazwy hostów i adresy oraz dodajesz adresy używanych gatewayów i serwerów NIS**.

Powinieneś się upewnić, że twój resolver w czasie testowania przy inicjacji wykorzystuje jedynie informacje z pliku *hosts*. Przykładowe pliki dostarczane wraz z oprogramowaniem DNS czy NIS mogą dawać dziwne rezultaty. Aby wszystkie aplikacje korzystały wyłącznie z */etc/hosts* przy poszukiwaniu adresu IP hosta, musisz dokonać edycji pliku */etc/host.conf*. Poprzedź znakiem komentarza (#) wszystkie linie zaczynające się od słowa kluczowego `order`, i wstaw wiersz:

```
order hosts
```

Konfiguracja resolvera jest dokładnie opisana w rozdziale 6, *Usługi nazewnicze i konfigurowanie resolvera*.

Plik *hosts* zawiera po jednym wpisie w wierszu, a każdy wpis składa się z adresu IP i nazwy hosta oraz opcjonalnej listy aliasów tej nazwy. Pola są oddzielone spacjami albo tabulatorami, a pole adresu musi zaczynać się w pierwszej kolumnie. Wszystko, co jest poprzedzone hashem, jest uznawane za komentarz i ignorowane.

* Pierwsza liczba w każdej podsieci to jej adres, a ostatnia to adres rozgłoszeniowy, a więc w podsieci można w rzeczywistości umieścić tylko 62 hosty.

** Adres serwera NIS jest potrzebny tylko wtedy, gdy używasz NYS Petera Erikssona. Inne implementacje NIS znajdują swoje serwery w czasie pracy za pomocą polecenia *ypbind*.

Nazwy hostów mogą być pełne lub względne dla domeny lokalnej. W przypadku **vale** wprowadziłbyś w pliku *hosts* pełną nazwę **vale.vbrew.com** oraz **vale**, aby host był znany zarówno pod nazwą oficjalną, jak i skróconą – lokalną.

Oto przykład, jak może wyglądać plik *hosts* dla wirtualnego browaru. Dodano dwie nazwy specjalne **vlager-if1** i **vlager-if2**, które zawierają adresy obu interfejsów używanych na hoście **vlager**:

```
#
# Plik hosts dla wirtualnego browaru/wirtualnej winiarni
#
# IP          FQDN                    aliasy
#
127.0.0.1     localhost
#
172.16.1.1    vlager.vbrew.com        vlager vlager-if1
172.16.1.2    vstout.vbrew.com        vstout
172.16.1.3    vale.vbrew.com          vale
#
172.16.2.1    vlager-if2
172.16.2.2    vbeaujolais.vbrew.com   vbeaujolais
172.16.2.3    vbardolino.vbrew.com    vbardolino
172.16.2.4    vchianti.vbrew.com      vchianti
```

Podobnie jak w przypadku adresów IP hosta, tak i dla numerów sieci po winieneś czasem używać również nazw symbolicznych. Dla tego plik *hosts* posiada bliźniaczy plik */etc/networks*, który odwzorowuje nazwy sieci na ich numery i odwrotnie. W wirtualnym browarze zainstalowalibyśmy następujący plik *networks*:*

```
# /etc/networks dla wirtualnego browaru
brew-net      172.16.1.0
wine-net      172.16.2.0
```

Konfigurowanie interfejsu dla IP

Zgodnie z tym, co napisaliśmy w rozdziale 4, *Konfigurowanie urządzeń szeregowych*, po skonfigurowaniu sprzętu musisz zadbać o to, aby oprogramowanie sieciowe jądra rozpoznawało urządzenia. Do skonfigurowania interfejsów sieciowych i zainicjowania tablicy rutingu stosuje się kilka poleceń. Zadania te zwykle są wykonywane ze skryptu inicjującego go sieć każdorazowo podczas uruchomienia systemu. Podstawowe narzędzia do tego celu to *ifconfig* (gdzie *if* jest skrótem od *interfejs*) i *route*.

Polecenie *ifconfig* jest używane do udostępniania interfejsu warstwie sieciowej jądra. Wymaga to przypisania adresu IP i zdefiniowania innych parametrów oraz aktywacji interfejsu, często nazywanej także „podniesieniem" interfejsu. Interfejs aktywny oznacza tutaj, że jądro będzie przez niego wysyłało i odbierało datagramy IP. Oto najprostsza procedura tego zadania:

```
ifconfig interfejs adres-IP
```

* Pamiętaj, że nazwy w pliku *networks* nie mogą kłócić się z nazwami z pliku *hosts*, gdyż niektóre programy mogłyby się dziwnie zachowywać.

Polecenie to przypisuje `adres-IP` do `interfejsu` i go aktywuje. Wszystkie pozostałe parametry są ustawiane na wartości domyślne. Na przykład domyślna maska sieci jest ustalana na podstawie klasy sieci, do której należy podany adres IP, czyli **255.255.0.0** dla klasy B. *ifconfig* jest opisane szczegółowo w podrozdziale *Wszystko o ifconfig*.

Polecenie *route* pozwala na dodanie lub usunięcie trasy z tablicy rutingu jądra. Można wywołać je następująco:

```
route [add|del] [-net|-host] przeznaczenie [if]
```

Argumenty `add` i `del` określają, czy należy dodać, czy też usunąć trasę do `przeznaczenia`. Argumenty `-net` i `-host` mówią poleceniu *route*, czy `przeznaczenie` to sieć, czy host (domyślnie, jeżeli nic nie podasz, przyjmowany jest host). Argument `if` jest opcjonalny i pozwala na podanie interfejsu sieciowego, do którego powinna zostać przekierowana dana trasa – jądro Linuksa rozsądnie zgaduje, jeżeli nie podasz tej informacji. Temat ten zostanie szczegółowo wyjaśniony w kolejnych podrozdziałach.

Interfejs pętli zwrotnej

W pierwszej kolejności aktywowany jest interfejs pętli zwrotnej:

```
# ifconfig lo 127.0.0.1
```

Może się zdarzyć, że zamiast adresu IP zobaczysz fikcyjną nazwę hosta **localhost**. *ifconfig* będzie szukać nazwy w pliku *hosts*, gdzie powinien znajdować się wpis wiążący tę nazwę z adresem **127.0.0.1**.

```
# Przyk□adowy wpis localhost w /etc/hosts
localhost       127.0.0.1
```

Aby obejrzeć konfigurację interfejsu, wywołujesz polecenie *ifconfig*, podając jako argument jedynie nazwę interfejsu.

```
$ ifconfig lo
lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        UP LOOPBACK RUNNING  MTU:3924  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        Collisions:0
```

Jak widzisz, interfejsowi pętli zwrotnej została przypisana maska sieci **255.0.0.0**, ponieważ adres **127.0.0.1** należy do klasy A.

Teraz w zasadzie możesz zacząć zabawę ze swoją minisiecią. Wciąż jednak brakuje wpisu w tablicy rutingu, mówiącego IP, że może używać tego interfejsu jako trasy do adresu **127.0.0.1**. Można go dodać następująco:

```
# route add 127.0.0.1
```

Znów możesz użyć **localhost** zamiast adresu IP, pod warunkiem, że wpisałeś go do pliku */etc/hosts*.

Następnie powinieneś sprawdzić, czy wszystko poprawnie działa, na przykład używając polecenia *ping*. Polecenie to sprawdza, czy podany adres jest rzeczywiście osiągalny, i mierzy opóźnienia występujące przy wysyłaniu datagramu na ten adres i z powrotem. Czas potrzebny do wykonania tego zadania jest często nazywany „czasem przewidzianym na transmisję i potwierdzenie przyjęcia” (ang. *round-trip time*):

```
# ping localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=255 time=0.4 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=255 time=0.4 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=255 time=0.4 ms
^C
--- localhost ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.4/0.4/0.4 ms
#
```

Kiedy *ping* zostanie wywołane w pokazany tu sposób, będzie wysyłało pakiety dopóty, dopóki użytkownik nie przerwie wykonywania polecenia. Znak `^C` pokazuje, gdzie nacisnęliśmy [CTRL+C].

W tym przykładzie widać, że pakiety są poprawnie dostarczane na adres **127.0.0.1**, a odpowiedź jest zwracana natychmiast. To znak, że prawidłowo skonfigurowałeś swój pierwszy interfejs sieciowy.

Jeśli wynik polecenia *ping* nie przypomina pokazanego w powyższym przykładzie, znaczy to, że masz kłopot. Sprawdź wszelkie odstępstwa – czy nie wskazują one, że jakieś pliki nie zostały poprawnie zainstalowane? Sprawdź, czy binaria *ifconfig* i *route*, których używasz, są kompatybilne z twoją wersją jądra, a przede wszystkim, czy jądro zostało skompilowane z obsługą sieci (powinieneś mieć katalog */proc/net*). Jeżeli otrzymasz komunikat o treści „Network unreachable”, prawdopodobnie zrobiłeś coś nie tak w poleceniu *route*. Sprawdź, czy użyłeś tego samego adresu, który podałeś w *ifconfig*.

Przedstawiona procedura powinna umożliwić ci korzystanie z aplikacji sieciowych na pojedynczym hoście. Po dodaniu wcześniej wspomnianych linii do skryptu inicjującego sieć i upewnieniu się, że zostanie on uruchomiony w czasie startu, możesz ponownie uruchomić swój komputer i wypróbować różne aplikacje. Na przykład *telnet localhost* powinno zrealizować połączenie *telnet* z twoim hostem, pokazując monit `login:`.

Jednak interfejs pętli zwrotnej jest przydatny nie tylko jako przykład w książkach o sieci czy do testowania oprogramowania, ale jest rzeczywiście używany przez niektóre aplikacje w czasie normalnej pracy*. Dlatego zawsze musisz go konfigurować bez względu na to, czy twoja maszyna jest podłączona do sieci, czy nie.

* Na przykład wszystkie aplikacje oparte na RPC wykorzystują interfejs pętli zwrotnej do rejestrowania się w demonie *portmapper* w czasie startu. Do aplikacji tych należą NIS i NFS.

Interfejsy Ethernet

Konfigurowanie interfejsu Ethernet jest prawie identyczne z konfigurowaniem interfejsu pętli zwrotnej. Wymaga jedynie wprowadzenia kilku dodatkowych parametrów, jeżeli używasz podziału na podsieci.

W wirtualnym browarze podzieliliśmy oryginalną sieć IP klasy B na podsieci o postaci sieci klasy C. Aby interfejs rozpoznał podział na podsieci, wywołanie *ifconfig* powinno być następujące:

```
# ifconfig eth0 vstout netmask 255.255.255.0
```

Polecenie to przypisuje interfejsowi *eth0* adres IP **vstout** (**172.16.1.2**). Gdybyśmy pominęli maskę sieci, *ifconfig* zredukowałoby maskę sieci na podstawie adresu IP i uzyskalibyśmy niepoprawną maskę postaci **255.255.0.0**. Teraz szybko sprawdzamy wynik:

```
# ifconfig eth0
eth0      Link encap 10Mps Ethernet HWaddr 00:00:C0:90:B3:42
          inet addr 172.16.1.2 Bcast 172.16.1.255 Mask 255.255.255.0
          UP BROADCAST RUNNING MTU 1500 Metric 1
          RX packets 0 errors 0 dropped 0 overrun 0
          TX packets 0 errors 0 dropped 0 overrun 0
```

Możesz zauważyć, że *ifconfig* automatycznie ustawia adres rozgłoszeniowy (pole `Bcast`) na typową wartość, która składa się z numeru sieci hosta i ustawionych wszystkich bitów w części numeru hosta. Także maksymalna jednostka transmisji (maksymalny rozmiar datagramów IP tworzonych dla interfejsu przez jądro) została ustawiona na maksymalny rozmiar pakietów Ethernet: 1500 bajtów. Zwykle będziesz używał wartości domyślnych, ale w razie potrzeby możesz je zmienić za pomocą specjalnych opcji, które zostaną opisane w podrozdziale *Wszystko o ifconfig*.

Tak jak przy konfiguracji interfejsu pętli zwrotnej, musisz teraz utworzyć wpis w tablicy rutingu mówiący jądru o sieci, która jest osiągalna przez *eth0*. W przypadku wirtualnego browaru możesz wywołać polecenie *route* następująco:

```
# route add -net 172.16.1.0
```

Z początku wygląda to nieco tajemniczo, ponieważ nie jest zupełnie jasne, jak *route* wykrywa interfejs, przez który ma przesyłać pakiety. Jednak cały zabieg jest raczej prosty: jądro sprawdza wszystkie interfejsy, które zostały do tej pory skonfigurowane, i porównuje adres docelowy (w tym przypadku **172.16.1.0**) z częścią sieciową adresu interfejsu (to znaczy wykonuje bitową logiczną operację AND na adresie interfejsu i masce sieci). Jedynym pasującym interfejsem jest *eth0*.

A do czego służy opcja *-net*? Jest ona potrzebna, ponieważ *route* może obsłużyć obie trasy: do sieci i do pojedynczego hosta (co widziałeś wcześniej przy **localhost**). Kiedy podamy adres w notacji kropkowej, *route* próbuje zgadnąć, czy jest to adres sieci czy hosta, patrząc na bity części hosta. Jeżeli w adresie część hosta jest zerowa, *route* zakłada, że chodzi o adres sieci – w przeciwnym razie *route* uzna je go za adres hosta. Dlatego *route* uznałoby, że **172.16.1.0** to adres hosta, a nie adres sieci, ponieważ nie wie, że zastosowaliśmy podział na podsieci. Musimy powiedzieć jawnie, że chodzi o sieć, a więc podajemy opcję *-net*.

Oczywiście wpisywanie polecenia *route* jest nużące i łatwo przy tym o pomyłkę. Wygodniejsze jest użycie nazw sieci, które zdefiniowaliśmy w */etc/networks*. Polecenie staje się bardziej czytelne i można nawet pominąć opcję *-net*, ponieważ *route* wie, że **172.16.1.0** oznacza sieć.

```
# route add brew-net
```

Teraz, gdy już masz za sobą podstawowe kroki konfiguracyjne, upewnj się, że interfejs Ethernet naprawdę działa poprawnie. Wybierz jakiś host ze swojej sieci, na przykład **vlager**, i napisz:

```
# ping vlager
PING vlager: 64 byte packets
64 bytes from 172.16.1.1: icmp_seq=0. time=11. ms
64 bytes from 172.16.1.1: icmp_seq=1. time=7. ms
64 bytes from 172.16.1.1: icmp_seq=2. time=12. ms
64 bytes from 172.16.1.1: icmp_seq=3. time=3. ms
^C
--- vstout.vbrew.com PING Statistics ---
4 packets transmitted, 4 packets received, 0
round-trip (ms) min/avg/max = 3/8/12
```

Jeżeli twój wynik się różni, coś jest nie tak. Jeżeli zauważysz, że współczynnik utraty pakietów ma jakąś nieprawdopodobną wartość, wskazuje to na problem sprzętowy, na przykład złe terminatory w przypadku kabla wspołosiowego lub ich brak.

Jeżeli nie uzyskasz w ogóle żadnych odpowiedzi, powinieneś sprawdzić konfigurację interfejsu za pomocą polecenia *netstat* (patrz podrozdział *Polecenie netstat* w tym rozdziale). Statystyka pakietów pokazana przez *ifconfig* powinna powiedzieć ci, czy jakieś pakiety zostały w ogóle wysłane przez interfejs. Jeżeli masz również dostęp do zdalnego hosta, powinieneś podejść do niego i sprawdzić statystyki interfejsu. W ten sposób możesz dokładnie stwierdzić, gdzie pakiety zostały zgubione. Ponadto za pomocą polecenia *route* powinieneś wyświetlić informacje o rutingu i zobaczyć, czy oba hosty mają poprawne wpisy dotyczące rutingu. *route* wyświetli pełną tablicę rutingu jądra, jeżeli zostanie wywołane bez argumentów (*-n* powoduje jedynie, że są drukowane adresy w postaci numerycznej, a nie nazwy hostów):

```
# route -n
Kernel routing table
Destination     Gateway   Genmask          Flags  Metric Ref Use Iface
127.0.0.1       *         255.255.255.255  UH     1      0   112 lo
172.16.1.0      *         255.255.255.0    U      1      0   10  eth0
```

Znaczenie poszczególnych pól zostanie wyjaśnione dalej w podrozdziale *Polecenie netstat*. Kolumna `Flags` zawiera listę znaczników ustawionych dla każdego interfejsu. `U` zawsze jest ustawione w przypadku aktywnych interfejsów, a `H` mówi, że adresem docelowym jest adres hosta. Jeżeli znacznik `H` jest ustawiony dla trasy, która miała być trasą do sieci, musisz ponownie wydać polecenie *route* z opcją *-net*. Aby dowiedzieć się, czy wprowadzona trasa jest w ogóle używana, sprawdź, czy wartość pola `Use` w drugiej kolumnie od końca zwiększa się pomiędzy wywołaniami polecenia *ping*.

Ruting przez gateway

W poprzednim podrozdziale omówiliśmy konfigurację hosta podłączonego do jednej sieci Ethernet. Często się jednak zdarza, że sieci są ze sobą połączone. Służą do tego gatewaye, które łączą po prostu dwie lub więcej sieci Ethernet, ale mogą także stanowić pomost do świata zewnętrznego, na przykład do Internetu. Aby wykorzystać gateway, musisz podać warstwie sieciowej dodatkowe informacje o rutingu.

Sieci Ethernet należące do wirtualnego browaru i wirtualnej winiarni są połączone takim właśnie gatewayem. Nosi on nazwę **vlager**. Zakładając, że **vlager** został już skonfigurowany, musimy po prostu dodać do tablicy rutingu **vstout** wpis, który poinformuje jądro, jak może dostać się przez **vlager** do wszystkich hostów w sieci winiarni. Odpowiednie wywołanie polecenia *route* zostało pokazane poniżej. Słowo kluczowe `gw` mówi, że następny argument oznacza gateway:

```
# route add wine-net gw vlager
```

Oczywiście dowolny host w sieci winiarni, z którym się chcesz połączyć, musi mieć wpis dotyczący trasy do sieci browaru. W przeciwnym razie będziesz w stanie jedynie wysłać dane z sieci browaru do sieci winiarni, ale hosty w sieci winiarni nie będą w stanie odpowiedzieć.

Przykład ten opisuje tylko gateway, który przekazuje pakiety pomiędzy dwoma wyizolowanymi sieciami Ethernet. Załóżmy teraz, że **vlager** ma także połączenie z Internetem (przyjmij, że przez dodatkowe łącze SLIP). W takim razie chcemy, aby datagramy adresowane do *dowolnego* miejsca poza siecią browaru były obsługiwane przez **vlager**. Można to zrobić, definiując ten gateway jako domyślny dla **vstout**:

```
# route add default gw vlager
```

Nazwa sieci **default** stanowi skrót dla adresu **0.0.0.0**, który oznacza trasę domyślną. Domyślna trasa pasuje do każdego adresu docelowego i będzie używana, jeżeli w tablicy rutingu nie ma dokładniejszej trasy. Nie musisz dodawać tej nazwy do pliku */etc/networks*, ponieważ jest ona wbudowana w polecenie *route*.

Jeżeli polecenie *ping* skierowane do hosta znajdującego się za jednym lub kilkoma gatewayami pokazuje, że dużo pakietów jest gubionych, może to oznaczać, że sieć jest zapchana. Gubienie pakietów nie wynika z wad technicznych, ale z tymczasowego przeciążenia hostów przekazujących, które powoduje opóźnienia, a nawet gubienie przychodzących pakietów.

Konfigurowanie gatewaya

Skonfigurowanie maszyny, która przekazuje pakiety pomiędzy dwoma sieciami Ethernet, jest dosyć proste. Wróćmy do hosta **vlager**, który jest wyposażony w dwie karty Ethernet, a każda z nich jest podłączona do jednej z dwóch sieci. Musisz jedynie skonfigurować oddzielnie obie karty i nadać im odpowiednie adresy IP oraz wyznaczyć trasy.

Dosyć przydatne jest dodanie informacji o obu interfejsach do pliku *hosts*, by mieć pod ręką łatwe do zapamiętania nazwy, co pokazano w poniższym przykładzie.

```
172.16.1.1   vlager.vbrew.com   vlager vlager-if1
172.16.2.1   vlager-if2
```

Aby skonfigurować te dwa interfejsy, należy wydać następujące polecenia:

```
# ifconfig eth0 vlager-if1
# route add brew-net
# ifconfig eth1 vlager-if2
# route add wine-net
```

Jeżeli te polecenia nie działają, sprawdź, czy jądro zostało skompilowane z włączoną opcją przekazywania IP (ang. *IP forwarding*). W tym celu upewnij się, czy pierwsza liczba w drugim wierszu pliku */proc/net/snmp* jest ustawiona na `1`.

Interfejs PLIP

Łącze PLIP używane do połączenia dwóch komputerów różni się nieco od Ethernetu. Łącza PLIP należą do łączy typu punkt-punkt, co oznacza, że na każdym końcu takiego łącza jest jeden host. Sieci typu Ethernet są nazywane sieciami *rozgłoszeniowymi*. Konfiguracja łączy punkt-punkt jest inna, ponieważ w odróżnieniu od sieci rozgłoszeniowych nie tworzą one własnej sieci.

PLIP to bardzo tanie i przenośne łącze pomiędzy komputerami. Jako przykład rozważmy komputer typu laptop należący do pracownika wirtualnego browaru. Komputer ten jest podłączony do hosta **vlager** przez łącze PLIP. Sam laptop nazywa się **vlite** i ma tylko jeden port równoległy. W czasie uruchamiania systemu port ten rejestruje się jako *plip1*. Aby uaktywnić łącze, musisz skonfigurować interfejs *plip1*, używając poniższych poleceń*:

```
# ifconfig plip1 vlite pointopoint vlager
# route add default gw vlager
```

Pierwsze polecenie konfiguruje interfejs mówiąc jądru, że jest to łącze punkt-punkt i że druga strona ma adres **vlager**. Drugie polecenie dodaje domyślną trasę, używając hosta **vlager** jako gatewaya. Do uaktywnienia łącza na hoście **vlager** niezbędne jest podobne polecenie *ifconfig* (wywołanie *route* nie jest potrzebne):

```
# ifconfig plip1 vlager pointopoint vlite
```

Zauważ, że interfejs *plip1* na hoście **vlager** nie potrzebuje oddzielnego adresu IP, ale można mu nadać również adres **172.16.1.1**. Łącza punkt-punkt nie obsługują bezpośrednio sieci, a więc interfejsy nie wymagają adresu. Jądro wykorzystuje informacje o interfejsie zawarte w tablicy routingu, aby uniknąć jakichś pomyłek**.

---

* Pisownia `pointopoint` nie jest błędna. Tak się po prostu pisze.

** Ostrożność nakazuje skonfigurować łącze PLIP czy SLIP dopiero wtedy, gdy w pełni skonfigurujesz wpisy w tablicy routingu dla kart Ethernet. W przeciwnym razie w niektórych starszych jądrach twoja trasa mogła się kończyć, wskazując na łącze punkt-punkt.

Skonfigurowaliśmy już ruting z laptopa do sieci browaru. Wciąż jednak nie mamy trasy z dowolnego hosta browaru do **vlite**. Dodawanie takiej trasy w tablicy rutingu każdego hosta jest wyjątkowo kłopotliwe. Trzeba byłoby tam wskazać, że **vlager** jest gatewayem dla **vlite**:

```
# route add vlite gw vlager
```

Dla tras tymczasowych lepszy jest ruting dynamiczny. Na każdym hoście w sieci można zainstalować demona rutingu *gated*, który będzie dynamicznie rozpowszechniał informacje o rutingu. Prostszym wyjściem jednak jest użycie *proxy ARP* (*Address Resolution Protocol*). Dzięki proxy ARP, **vlager** będzie odpowiadał na każde zapytanie ARP o **vlite**, wysyłając własny adres Ethernet. Wszystkie pakiety dla **vlite** będą docierały do **vlagera**, który następnie będzie je przekazywał do laptopa. Do proxy ARP powrócimy w podrozdziale *Sprawdzanie tablic ARP*.

Obecna wersja *net-tools* zawiera narzędzie o nazwie *plipconfig* pozwalające na ustawienie odpowiednich parametrów czasowych PLIP. IRQ dla portu równoległego można ustawić za pomocą polecenia *ifconfig*.

Interfejsy SLIP i PPP

Choć łącza SLIP i PPP są, tak jak PLIP, jedynie prostymi łączami typu punkt-punkt, można o nich powiedzieć dużo więcej. Zwykle ustanowienie połączenia SLIP wymaga zadzwonienia do drugiej strony przez modem i ustawienia trybu SLIP dla łącza szeregowego. Z PPP korzysta się podobnie. SLIP i PPP omawiamy dokładniej w rozdziałach 7, *IP łącza szeregowego*, i 8, *Protokół punkt-punkt*.

Interfejs fikcyjny (ang. *dummy interface*)

Interfejs fikcyjny (ang. *dummy interface*) jest nieco egzotyczny, ale jednak przydatny. Jego główną zaletą w przypadku pojedynczych hostów i komputerów posiadających jedynie podłączenie do sieci IP jest łącze komutowane. W rzeczywistości przeważnie obsługuje pojedyncze hosty.

Problem z pojedynczymi hostami polega na tym, że mają one aktywne tylko jedno urządzenie sieciowe – interfejs pętli zwrotnej, któremu zwykle jest przypisywany adres **127.0.0.1**. Czasami jednak musisz wysłać dane na „oficjalny" adres IP hosta lokalnego. Na przykład załóżmy, że laptop **vlite** został chwilowo odłączony od sieci. Aplikacja na **vlite** może teraz chcieć wysłać dane do innej aplikacji na tym samym hoście. Sprawdzenie **vlite** w pliku */etc/hosts* daje adres IP **172.16.1.65**, a więc aplikacja próbuje wysyłać dane na taki adres. Ponieważ interfejs pętli zwrotnej jest obecnie jedynym aktywnym interfejsem w tym komputerze, jądro nie ma pojęcia, że adres **172.16.1.65** tak naprawdę odnosi się do tej samej maszyny! W konsekwencji jądro odrzuca datagram i zwraca do aplikacji komunikat o błędzie.

Tu właśnie przydaje się interfejs fikcyjny. Rozwiązuje on problem wykorzystując interfejs pętli zwrotnej. W przypadku **vlite**, po prostu nadajesz interfejsowi adres **172.16.1.65** i dodajesz trasę, tak by na niego wskazywała. Każdy datagram przezna-

czony dla adresu **172.16.1.65** będzie od tej chwili dostarczony lokalnie. Oto poprawne wywołanie*:

```
# ifconfig dummy vlite
# route add vlite
```

Alias IP

Nowe jądra obsługują funkcję, która może zastąpić interfejs fikcyjny i pełnić inne użyteczne role. *Alias IP* pozwala na skonfigurowanie wielu adresów IP na jednym urządzeniu fizycznym. W najprostszym przypadku można na przykład zrealizować interfejs fikcyjny. Wystarczy skonfigurować adres hosta jako alias dla interfejsu pętli zwrotnej i możesz zupełnie zrezygnować z interfejsu fikcyjnego. W bardziej złożonych zastosowaniach mógłbyś skonfigurować swój host tak, by wyglądał jak inne hosty, każdy o swoim własnym adresie IP. Konfiguracja taka jest czasem nazywana tworzeniem hostów wirtualnych, choć technicznie jest również używana w wielu innych celach**.

Aby skonfigurować alias dla interfejsu, musisz najpierw sprawdzić, czy jądro zostało skonfigurowane z obsługą aliasów IP (sprawdź, czy masz plik */proc/net/ip_alias*; jeżeli nie – trzeba ponownie skompilować jądro). Konfiguracja aliasu IP przebiega tak samo jak konfiguracja normalnego urządzenia sieciowego. Jedyna różnica polega na użyciu specjalnej nazwy wskazującej, że jest to alias. Na przykład:

```
# ifconfig lo:0 172.16.1.1
```

To polecenie utworzy alias o adresie `172.16.1.1` dla interfejsu pętli zwrotnej. Aliasy IP są oznaczane przez dodanie `:n` do rzeczywistej nazwy urządzenia sieciowego, gdzie „n" jest liczbą całkowitą. W naszym przykładzie tworzymy alias o numerze 0 dla urządzenia sieciowego `lo`. Dzięki numeracji pojedyncze urządzenie fizyczne może obsłużyć wiele aliasów.

Każdy alias może być traktowany jako oddzielne urządzenie i z punktu widzenia oprogramowania IP jądra tak właśnie jest. Będzie jednak współdzielił sprzęt z innym interfejsem.

Wszystko o ifconfig

Polecenie *ifconfig* ma dużo więcej parametrów, niż opisaliśmy do tej pory. Typowe wywołanie wygląda tak:

```
ifconfig interfejs [adres [parametry]]
```

* Urządzenie fikcyjne nosi nazwę *dummy0*, jeżeli załadowałeś jego sterownik jako moduł, a nie wbudowałeś w jądro. Numeracja jest potrzebna, ponieważ możesz załadować wiele modułów i mieć więcej niż jedno urządzenie fikcyjne

** Używanie aliasów IP bardziej poprawnie jest nazywane tworzeniem hostów wirtualnych w warstwie sieciowej. W świecie WWW i STMP bardziej popularne jest tworzenie hostów wirtualnych używanych w warstwie aplikacji, gdzie ten sam adres IP jest używany dla każdego hosta wirtualnego, ale przy każdym żądaniu warstwy aplikacji jest podawana inna nazwa hosta. Usługi takie jak FTP nie są w stanie działać w ten sposób i wymagają hostów wirtualnych w warstwie sieciowej.

Oczywiste jest, że `interfejs` to nazwa interfejsu, a `adres` to nazwa adresu IP przypisanego interfejsowi. Może być to adres w postaci liczbowej lub nazwa, którą *ifconfig* odnajdzie w pliku */etc/hosts*.

Gdybyśmy wywołali *ifconfig* tylko z nazwą interfejsu, zobaczylibyśmy konfigurację interfejsu. Przy wywołaniu bez parametrów *ifconfig* wyświetla wszystkie interfejsy, jakie masz do tej pory skonfigurowane. Opcja *-a* wymusza również pokazanie interfejsów nieaktywnych. Przykładowe wywołanie dla interfejsu Ethernet *eth0* może wyglądać następująco:

```
# ifconfig eth0
eth0      Link encap 10Mbps Ethernet HWaddr 00:00:C0:90:B3:42
           inet addr 172.16.1.2 Bcast 172.16.1.255 Mask 255.255.255.0
          UP BROADCAST RUNNING MTU 1500 Metric 0
          RX packets 3136 errors 217 dropped 7 overrun 26
          TX packets 1752 errors 25 dropped 0 overrun 0
```

Pola `MTU` i `Metric` pokazują aktualne wartości MTU i metryki interfejsu. Metryka jest tradycyjnie używana przez niektóre systemy operacyjne do obliczenia kosztu trasy. Linux nie korzysta z tej wartości, ale definiuje ją dla zachowania kompatybilności.

Wiersze `RX` i `TX` pokazują, ile pakietów zostało pomyślnie odebranych i wysłanych, ile wystąpiło błędów, ile pakietów zostało pominiętych (prawdopodobnie ze względu na brak pamięci), a ile zostało zgubionych ze względu na przeciążenie. Przeciążenia odbiorcy występują zwykle wtedy, gdy pakiety nadchodzą szybciej niż jądro jest w stanie obsłużyć ostatnie przerwanie. Znaczniki pokazywane przez *ifconfig* z grubsza odpowiadają nazwom opcji wiersza poleceń, które omówimy dalej.

Poniżej przedstawiamy listę parametrów rozpoznawanych przez *ifconfig* z odpowiednimi nazwami znaczników. Opcje, które włączają funkcję, pozwalają również ją wyłączyć, jeśli przed opcją umieścimy znak minus (-).

`up`

: Ta opcja udostępnia interfejs warstwie IP. Opcja jest domyślna w momencie podania w wierszu poleceń `adresu`. Może być także użyta do ponownego włączenia interfejsu, który został tymczasowo zamknięty za pomocą opcji `down`.

: Z tą opcją związane są znaczniki *UP* i *RUNNING*.

`down`

: Ta opcja oznacza, że interfejs jest niedostępny dla warstwy IP. W rzeczywistości odcina cały ruch IP do interfejsu. Zauważ, że ta opcja usunie automatycznie także wszystkie wpisy w tablicy routingu, które wykorzystują dany interfejs.

`netmask` *`maska`*

: Ta opcja określa maskę sieci używaną przez interfejs. Może być ona podana w postaci 32-bitowej liczby szesnastkowej poprzedzonej 0x albo w postaci liczb dziesiętnych oddzielonych kropkami. Choć postać liczb dziesiętnych jest popularniejsza, często dużo łatwiej jest pracować z notacją szesnastkową. Maski sieci są w gruncie rzeczy binarne i łatwiej dokonać konwersji z zapisu binarnego na szesnastkowy, niż z binarnego na dziesiętny.

`pointopoint` *`adres`*

Ta opcja jest używana na przyłączach IP punkt-punkt łączących tylko dwa hosty. Jest potrzebna na przykład do skonfigurowania interfejsów SLIP i PPP. Jeżeli zostanie ustawiony adres punkt-punkt, *ifconfig* wyświetli znacznik *POINTOPOINT*.

`broadcast` *`adres`*

Adres rozgłoszeniowy jest zwykle złożony z adresu sieci i wszystkich bitów ustawionych w adresie hosta. Niektóre implementacje IP (systemy pochodzące na przykład z BSD 4.2) wykorzystują inny schemat, w którym wszystkie bity w części hosta są zerowane. Opcja *broadcast* przystosowuje się do tych dziwnych środowisk. Jeżeli adres rozgłoszeniowy został określony, *ifconfig* wyświetla znacznik *BROADCAST*.

`irq`

Ta opcja pozwala ustalić IRQ używane przez zadane urządzenia. Jest to przydatne zwłaszcza w łączach PLIP, ale także przy niektórych kartach Ethernet.

`metric` *`liczba`*

Ta opcja może być użyta do przypisania metryki we wpisie utworzonym dla interfejsu w tablicy rutingu. Metryka jest używana przez protokół rutowania RIP (*Routing Information Protocol*) do tworzenia tablic rutingu dla sieci*. Domyślna metryka używana przez *ifconfig* ma wartość zero. Jeżeli nie korzystasz z demona RIP, nie potrzebujesz w ogóle tej opcji. A nawet jeżeli go używasz, rzadko będziesz musiał zmieniać wartość metryki.

`mtu` *`bajty`*

W ten sposób ustawia się maksymalną jednostkę transmisji, która określa maksymalną liczbę oktetów, jaką interfejs jest w stanie obsłużyć w jednym ruchu. W przypadku Ethernetu domyślna wartość MTU wynosi 1500 (największy dopuszczalny rozmiar dla pakietu Ethernet). W przypadku interfejsów SLIP jest to 296 (nie ma ograniczeń MTU w łączach SLIP – ta wartość jest po prostu pewnym kompromisem).

`arp`

Ta opcja jest właściwa dla sieci rozgłoszeniowych, takich jak Ethernet, lub dla radiapakietowego. Włącza ona protokół rozwiązywania adresów (ARP), używany do znajdowania fizycznych adresów hostów podłączonych do sieci. W sieciach rozgłoszeniowych użycie tego protokołu jest domyślne. Jeżeli ARP jest wyłączony, *ifconfig* wyświetla znacznik *NOARP*.

`-arp`

Ta opcja wyłącza użycie ARP na interfejsie.

---

* RIP wybiera optymalną trasę do zadanego hosta na podstawie „długości" drogi. Jest ona obliczana przez zsumowanie pojedynczych metryk na każdym łączu host-host. Domyślnie hop ma wartość 1, ale może być to całkowita wartość dodatnia mniejsza od 16. Długość trasy o wartości 16 odpowiada nieskończoności. Takie trasy są uznawane za bezużyteczne. Parametr *metric* ustala koszt hopa, który jest następnie rozgłaszany przez demona rutingu.

`promisc`

Ta opcja przełącza interfejs w tryb przechwytywania pakietów (ang. *promiscuous mode*). W sieciach rozgłoszeniowych oznacza to, że interfejs odbiera wszystkie pakiety, bez względu na to, czy są one dla niego przeznaczone, czy też nie. Pozwala to na analizę ruchu za pomocą filtrów pakietów i tym podobnych narzędzi, co jest nazywane także *snoopingiem Ethernetu*. Zwykle jest to dobra technika wykrywania problemów z siecią, które inną metodą byłyby trudne do znalezienia. Narzędzia takie jak *tcpdump* opierają się na tym trybie interfejsu.

Z drugiej strony opcja ta pozwala włamywaczom na robienie brzydkich rzeczy, na przykład na przeglądanie pakietów twojej sieci w poszukiwaniu haseł. Możesz się zabezpieczyć przed tego typu atakiem, zabraniając komukolwiek włączać komputery do twojej sieci Ethernet. Możesz także używać bezpiecznych protokołów uwierzytelniania, takich jak Kerberos czy pakiet secure shell*. Opcji tej odpowiada znacznik *PROMISC*.

`-promisc`

Ta opcja wyłącza tryb przechwytywania pakietów.

`allmulti`

Adresy grupowe (ang. *multicast addresses*) mają wiele wspólnego z adresami rozgłoszeniowymi Ethernet, z tym wyjątkiem, że nie implikują automatycznego kierowania do pakietów wszystkich członków sieci. Pakiety wysłane na adres grupowy dostają tylko te osoby, które ustawiły ich odbieranie. Jest to przydatne w takich zastosowaniach, jak wideokonferencje oparte na sieci Ethernet czy audio w sieci, gdzie tylko zainteresowani odbierają dane pakiety. Adresowanie grupowe jest obsługiwane przez większość sterowników Ethernet, aczkolwiek nie przez wszystkie. Gdy opcja ta jest włączona, interfejs odbiera i przekazuje pakiety grupowe do przetwarzania. Opcji tej odpowiada znacznik *ALLMULTI*.

`-allmulti`

Ta opcja wyłącza adresy grupowe.

Polecenie netstat

netstat jest przydatnym narzędziem do sprawdzania konfiguracji sieci i jej działania. W gruncie rzeczy jest to zestaw kilku połączonych ze sobą narzędzi. W kolejnych podrozdziałach omawiamy każdą z funkcji polecenia.

Wyświetlanie tablicy rutingu

Kiedy wywołasz *netstat* z opcją *-r*, wyświetli ono tablicę rutingu jądra w postaci podobnej jak polecenie *route*. Na hoście **vstout** wynik wygląda następująco:

```
# netstat -nr
Kernel IP routing table
Destination   Gateway     Genmask         Flags  MSS   Window   irtt   Iface
127.0.0.1     *           255.255.255.255 UH     0     0        0      lo
```

* *ssh* znajduje się w katalogu */pub/ssh* pod adresem **ftp.cs.hut.fi.**

```
172.16.1.0    *           255.255.255.0   U     0    0      0    eth0
172.16.2.0    172.16.1.1  255.255.255.0   UG    0    0      0    eth0
```

Opcja *-n* powoduje, że *netstat* wyświetla adresy w postaci numerów IP, a nie symbolicznych nazw hostów i sieci. Opcja ta jest szczególnie przydatna, jeżeli chcesz uniknąć szukania adresów w sieci (tzn. na serwerach DNS albo NIS).

Druga kolumna wyniku polecenia *netstat* pokazuje gateway, na który wskazuje dany wpis. Jeżeli gateway nie jest używany, wyświetlana jest gwiazdka. Trzecia kolumna pokazuje maskę sieci dla danej trasy. Gdy podamy adres IP, dla którego chcemy znaleźć odpowiednią trasę, jądro przegląda kolejne wpisy w tablicy rutingu i wykonuje bitowo logiczną operację AND na adresie i masce sieci, porównując adres docelowy z trasą.

Czwarta kolumna zawiera następujące znaczniki opisujące trasę:

`G` Trasa przez gateway.

`U` Interfejs, który ma być użyty, jest aktywny.

`H` Przez tę trasę można dostać się tylko do jednego hosta. Na przykład znacznik ten występuje w przypadku wpisu dla pętli zwrotnej **127.0.0.1**.

`D` Ta trasa jest tworzona dynamicznie. Znacznik jest ustawiany, jeżeli wpis w tablicy został stworzony przez demona rutingu, na przykład *gated*, lub przez komunikat przekierowania ICMP (zobacz podrozdział *Internetowy protokół komunikatów kontrolnych (ICMP)* w rozdziale 2).

`M` Ten znacznik jest ustawiony, jeżeli wpis w tablicy został zmodyfikowany przez komunikat przekierowania ICMP.

`!` Ta trasa została odrzucona i datagramy do niej skierowane będą gubione.

Kolejne trzy kolumny pokazują `MSS`, `Window` i `irtt`, czyli zmienne dotyczące połączeń TCP zrealizowanych w oparciu o daną trasę. `MSS` to maksymalny rozmiar segmentu (ang. *maximum segment size*); jest to rozmiar największego datagramu, jaki jądro może zbudować i wysłać tą trasą. `Window` to maksymalna liczba danych, jaką system przyjmie jednorazowo ze zdalnego hosta. Skrót `irtt` pochodzi od słów *initial round trip time* (wstępny czas przewidywany na transmisję i potwierdzenie przyjęcia). Protokół TCP zapewnia niezawodność dostarczania danych pomiędzy hostami, ponieważ ponownie wysyła datagram, jeżeli został on zgubiony. Pilnuje też licznika mierzącego czas potrzebny na dostarczenie datagramu na drugi koniec i odebranie potwierdzenia. Na podstawie tego licznika wie, po jakim czasie należy dokonać ewentualnej ponownej transmisji datagramu. Proces ten jest nazywany czasem przewidywanym na transmisję i potwierdzenie przyjęcia (ang. *round-trip time*). Wstępny czas przewidywany na transmisję i potwierdzenie przyjęcia to wartość, jakiej protokół TCP używa, kiedy po raz pierwszy realizuje połączenie. W większości typów sieci domyślna wartość jest poprawna, ale w przypadku niektórych wolnych sieci, jak na przykład w pewnych typach sieci amatorskiego radia pakietowego, czas ten jest zbyt krótki i powoduje niepotrzebne retransmisje. Wartość `irtt` może być ustawiona za pomocą polecenia *route*. Wartości zero w tych polach oznaczają, że będzie używana wartość domyślna.

No i ostatnie pole pokazuje interfejs sieciowy używany dla danej trasy.

Wyświetlanie statystyk interfejsu

Wywołanie *netstat* z opcją *-i* wyświetla statystyki obecnie skonfigurowanych interfejsów sieciowych. Jeżeli zostanie podana również opcja *-a*, wyświetlane są *wszystkie* interfejsy obecne w jądrze, a nie tylko te obecnie skonfigurowane. Na hoście **vstout** wynik polecenia *netstat* będzie wyglądał następująco:

```
# netstat -i
Kernel Interface table
Iface MTU Met RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP TX-OVR Flags
lo      0   0 3185       0      0      0  3185      0      0      0   BLRU
eth0 1500   0 972633    17     20    120 628711   217      0      0   BRU
```

Pola `MTU` i `Met` pokazują aktualnie ustalone wartości MTU i metryki dla danego interfejsu. Kolumny `RX` i `TX` pokazują, ile pakietów zostało odebranych albo wysłanych bezbłędnie (`RX-OK`/`TX-OK`), albo uszkodzonych (`RX-ERR`/`TX-ERR`), ile pakietów zostało zgubionych (`RX-DRP`/`TX-DRP`) oraz ile zostało zgubionych z powodu przeciążenia (`RX-OVR`/`TX-OVR`).

Ostatnia kolumna pokazuje znaczniki, które zostały ustawione dla danego interfejsu. Litery te są skróconą wersją długich nazw znaczników wyświetlanych dla konfiguracji interfejsu przez *ifconfig*:

`B` Został ustawiony adres rozgłoszeniowy.

`L` Ten interfejs to urządzenie pętli zwrotnej.

`M` Odbierane są wszystkie pakiety (tryb przechwytywania).

`O` ARP dla interfejsu jest wyłączony.

`P` Jest to połączenie punkt-punkt.

`R` Interfejs działa.

`U` Interfejs jest aktywny.

Wyświetlanie połączeń

netstat obsługuje zestaw opcji do wyświetlania aktywnych lub pasywnych gniazd. Opcje *-t*, *-u*, *-w* i *-x* pokazują aktywne połączenia TCP, UDP, RAW i gniazda Uniksa. Jeżeli ponadto podasz opcję *-a*, gniazda oczekujące na połączenie (tzn. nasłuchujące) także zostaną wyświetlone. W wyniku zobaczysz listę wszystkich serwerów, które aktualnie działają w twoim systemie.

Wywołanie *netstat -ta* na hoście **vlager** da następujący wynik:

```
$ netstat -ta
Active Internet Connections
Proto Recv-Q Send-Q Local Address    Foreign Address    (State)
tcp        0      0 *:domain         *:*                 LISTEN
tcp        0      0 *:time           *:*                 LISTEN
tcp        0      0 *:smtp           *:*                 LISTEN
tcp        0      0 vlager:smtp      vstout:1040         ESTABLISHED
tcp        0      0 *:telnet         *:*                 LISTEN
tcp        0      0 localhost:1046   vbardolino:telnet   ESTABLISHED
tcp        0      0 *:chargen        *:*                 LISTEN
tcp        0      0 *:daytime        *:*                 LISTEN
tcp        0      0 *:discard        *:*                 LISTEN
```

```
tcp        0      0 *:echo        *:*                LISTEN
tcp        0      0 *:shell       *:*                LISTEN
tcp        0      0 *:login       *:*                LISTEN
```

Widać, że większość serwerów po prostu oczekuje na nadchodzące połączenia. Jednak czwarta linia pokazuje przychodzące połączenie SMTP z hosta **vstout**, a szósta linia mówi, że istnieje wychodzące połączenie *telnet* do hosta **vbardolino***.

Użycie samej opcji *-a* wyświetli wszystkie gniazda ze wszystkich rodzin.

Sprawdzanie tablic ARP

W pewnych sytuacjach warto obejrzeć lub zmienić zawartość tablic ARP jądra. Przydaje się to, kiedy na przykład podejrzewasz, że zdublikowany adres internetowy jest powodem sporadycznych problemów z siecią. Narzędzie *arp* zostało stworzone do użycia w tego typu sytuacjach. Opcje wiersza poleceń *arp* są następujące:

```
arp [-v] [-t typhw] -a [nazwahosta]
arp [-v] [-t typhw] -s nazwahosta hwadres
arp [-v] -d nazwahosta [nazwahosta...]
```

Wszystkie argumenty `nazwahosta` mogą mieć postać symbolicznych nazw hostów lub adresów IP w notacji kropkowej.

Pierwsze wywołanie wyświetla wpis ARP dla podanego adresu IP lub nazwy hosta albo wszystkich hostów, jeżeli nie zostanie podana `nazwahosta`. Na przykład wywołanie *arp* na hoście **vlager** może pokazać coś takiego:

```
# arp -a
IP address    HW type           HW address
172.16.1.3    10Mbps Ethernet   00:00:C0:5A:42:C1
172.16.1.2    10Mbps Ethernet   00:00:C0:90:B3:42
172.16.2.4    10Mbps Ethernet   00:00:C0:04:69:AA
```

Są to adresy Ethernet hostów **vlager**, **vstout** i **vale**.

Używając opcji *-t*, możesz ograniczyć wyświetlanie do zadanego typu karty. Może to być `ether`, `ax25` albo `pronet`, czyli odpowiednio Ethernet 10 Mb/s, AMPR AX.25 i token ring IEEE 802.5.

Opcja *-s* jest używana do dodawania na stałe w tablicy ARP adresu ethernetowego `nazwyhosta`. Argument `hwadres` określa adres sprzętowy, który domyślnie powinien być adresem Ethernet podanym w postaci sześciu liczb szesnastkowych oddzielonych dwukropkami. Za pomocą opcji *-t* możesz również ustawić adresy sprzętowe dla innych typów urządzeń.

Zapytania ARP o zdalny host nie udają się czasem z różnych powodów. Na przykład gdy sterownik ARP jest błędny lub inny host w sieci błędnie identyfikuje

* Na podstawie numerów portów możesz stwierdzić, czy połączenie jest wychodzące. Numer portu pokazany dla hosta *wywołującego* będzie zawsze liczbą całkowitą. Na hoście, z którym jest realizowane połączenie, będzie używany port dobrze znanej usługi i *netstat* pokaże symboliczną nazwę, na przykład `smtp`, która jest zdefiniowana w pliku */etc/services*.

się z adresem IP innego hosta. Problem ten wymaga ręcznego dodania adresu IP do tablicy ARP. Adresy IP na sztywno (ręcznie) wpisane w tablicy ARP są również (bardzo drastycznym) zabezpieczeniem przed tymi hostami w twojej sieci, które udają, że są kim innym.

Wywołanie *arp* z opcją *-d* powoduje usunięcie wszystkich wpisów ARP dotyczących danego hosta. W ten sposób można wymusić na interfejsie ponowną próbę uzyskania adresu Ethernet związanego z danym adresem IP. Jest to przydatne, gdy błędnie skonfigurowany system rozgłosi złą informację ARP (oczywiście musisz najpierw przekonfigurować błędny host).

Wspomniana powyżej opcja *-s* może być używana do implementacji serwera *proxy* ARP. Jest to specjalna technika, dzięki której host, powiedzmy **gate**, działa jako gateway dla innego hosta **fnord** udając, że oba adresy odnoszą się do tego samego hosta **gate**. W tym celu rozgłasza wpis ARP dla **fnord** wskazujący na jego własny interfejs Ethernet. Teraz gdy host wyśle zapytanie ARP o **fnord**, **gate** zwróci odpowiedź zawierającą jego własny adres Ethernet. Pytający host wyśle wszystkie datagramy do **gate**, który z kolei posłusznie przekaże je do **fnord**.

Taki mechanizm może być potrzebny, gdy chcesz się dostać do **fnord** z komputera DOS-owego z niepoprawną implementacją TCP, nie zbyt dobrze rozumiejącą ruting. Gdy użyjesz proxy ARP, maszyna DOS-owa będzie widziała **fnord** tak, jak by był on w lokalnej podsieci, a więc nie będzie musiała rutować pakietów przez gateway.

Proxy ARP przydaje się też, gdy jeden z twoich hostów działa tymczasowo jako gateway dla innego hosta, na przykład połączonego przez łącze komutowane. W jednym z poprzednich przykładów spotkaliśmy się z laptopem **vlite**, który był czasem podłączony do **vlagera** przez łącze PLIP. Oczywiście protokół zadziała tylko wtedy, jeżeli adres hosta, dla którego chcesz realizować proxy ARP, znajduje się w tej samej podsieci IP co gateway. **vstout** może pełnić rolę proxy ARP dla dowolnego hosta w podsieci browaru (**172.16.1.0**), ale nie może dla hostów w podsieci winiarni (**172.16.2.0**).

Poniżej pokazujemy poprawne wywołanie tworzące proxy ARP dla **fnord**. Oczywiście podany adres Ethernet musi być adresem **gate'a**:

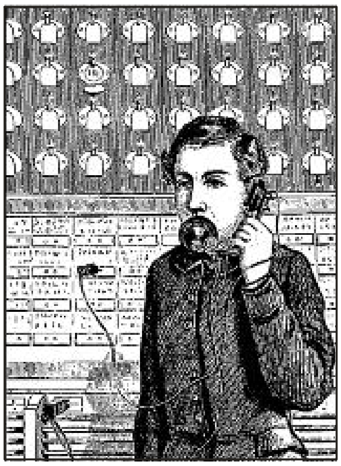
```
# arp -s fnord 00:00:c0:a1:42:e0 pub
```

Wpis proxy ARP może zostać usunięty przez ponowne wywołanie:

```
# arp -d fnord
```

6

Usługi nazewnicze i konfigurowanie resolvera



Jak powiedzieliśmy w rozdziale 2, *Wybrane problemy sieci TCP/IP*, w sieci TCP/IP mogą funkcjonować różne schematy konwersji nazw na adresy. Najprostszym rozwiązaniem jest tablica hostów zapisana w pliku */etc/hosts*. Sprawdza się ona jedynie w małych sieciach LAN, które są pod opieką jednego administratora albo nie obsługują ruchu IP do świata zewnętrznego. Format pliku *hosts* został już opisany w rozdziale 5, *Konfigurowanie sieci TCP/IP*.

Idąc dalej, do zamiany nazw hostów na adresy IP możesz użyć usługi z BIND (*Berkeley Internet Name Domain Service* – internetowe usługi nazewnicze domen z Berkeley). Konfigurowanie BIND-a bywa przykre, ale jak już to zrobisz, łatwo wprowadzisz każdą zmianę w topologii sieci. W Linuksie, jak i w wielu innych systemach uniksowych, usługi nazewnicze są obsługiwane przez program o nazwie *named*. Przy uruchamianiu ładuje on zestaw głównych plików do swojej pamięci wewnętrznej i czeka na zapytania od lokalnych lub zdalnych procesów użytkownika. Istnieją różne sposoby skonfigurowania BIND-a i nie wszystkie wymagają uruchamiania serwera nazw na każdym hoście.

Chcielibyśmy, aby ten rozdział był czymś więcej niż pobieżnym szkicem na temat DNS-u i obsługi serwera nazw. Informacje tu podane powinny wystarczyć, jeżeli masz małą sieć lokalną i połączenie z Internetem. Najświeższe informacje znajdziesz w dokumentacji zamieszczonej w pakiecie źródłowym BIND, który zawiera strony podręcznika elektronicznego, uwagi do danej wersji oraz *Podręcznik operatora BIND* (*BIND Operator's Guide* – BOG). Nie pozwól, by ta nazwa cię odstraszyła. W praktyce jest to bardzo użyteczny dokument. Pełniejszy opis DNS-u i związanych z nim zagadnień możesz znaleźć w książce *DNS and BIND* autorstwa Paula Albitza i Cricketa Liu (wyd. pol.: *DNS i BIND*, Wydawnictwo RM, Warszawa 1999), która stanowi doskonałe źródło informacji na ten temat. Odpowiedzi na pytania na temat DNS-u możesz znaleźć w wiadomościach grupy dyskusyjnej *comp.protocols.tcp-ip.domains*.

Szczegóły techniczne DNS-u są zdefiniowane w następujących dokumentach RFC: 1033, 1034 i 1035.

Biblioteka resolvera

Określenie *resolver* nie odnosi się do jakiejś szczególnej aplikacji, ale do biblioteki resolvera. Jest to zbiór funkcji, które można znaleźć w standardowej bibliotece C. Podstawowe procedury to *gethostyname(2)* i *gethostbyaddr(2)*, poszukujące adresów IP związanych z daną nazwą hosta i odwrotnie. Mogą być skonfigurowane tak, aby po prostu szukać informacji w pliku *hosts*, albo zadawać zapytania do serwerów nazw DNS, albo też korzystać z bazy danych *hosts* NIS-a.

Funkcje resolvera odczytują pliki konfiguracyjne w momencie, gdy są wywoływane. Na ich podstawie ustalają, którą bazę danych zapytać i w jakiej kolejności, oraz dowiadują się innych szczegółów na temat konfiguracji środowiska. Starsza standardowa biblioteka, *libc*, w Linuksie wykorzystywała plik */etc/host.conf* jako główny plik konfiguracyjny, ale wersja 2. standardowej biblioteki GNU, *glibc*, wykorzystuje plik */etc/nsswitch.conf*. Opiszemy kolejno każdy z nich, ponieważ oba są powszechnie używane.

Plik host.conf

Plik */etc/host.conf* mówi funkcjom resolvera ze starszej biblioteki standardowej w Linuksie, jakich usług używać i w jakiej kolejności.

Opcje w pliku *host.conf* trzeba umieszczać w oddzielnych wierszach. Pola mogą być oddzielone białymi znakami (spacjami lub tabulatorami). Znak hasha (#) oznacza linię z komentarzem. Dostępne są następujące opcje:

`order`
: Ta opcja określa kolejność, w jakiej wypróbowane są usługi. Dopuszczalne opcje to: `bind` dla zapytań serwera nazw, `hosts` dla sprawdzania pliku */etc/hosts* i `nis` dla zapytań NIS. Można podać jedną opcję lub wszystkie. Kolejność przepytywania (sprawdzania) usług zależy od uporządkowania opcji w wierszu.

`multi`
: `multi` może posiadać opcje `on` lub `off`. Określa, czy host wpisany do pliku */etc/hosts* może mieć kilka adresów IP. Domyślna wartość to `off`. Znacznik nie ma wpływu na zapytania do DNS-u czy NIS-a.

`nospoof`
: Jak wyjaśnimy dalej w podrozdziale *Wyszukiwanie odwrotne*, DNS pozwala na znalezienie nazwy hosta związanej z danym adresem IP za pomocą domeny **in-addr.arpa**. Próbę dostarczenia przez serwery nazw fałszywej nazwy hosta nazywa się *spoofingiem*. Aby się przed tym obronić, resolver można skonfigurować tak, żeby sprawdzał, czy oryginalny adres IP faktycznie jest związany z uzyskaną nazwą hosta. Jeżeli nie, nazwa jest odrzucana i zwracany jest błąd. Zachowanie to jest włączane przez ustawienie `nospoof`.

`alert`

Ta opcja przyjmuje parametry `on` lub `off`. Jeżeli jest włączona, próby spoofingu skończą się tym, że resolver zapisze komunikat za pomocą funkcji *syslog*.

`trim`

Jako argument tej opcji występuje nazwa domeny, usuniętej z nazw hostów przed rozpoczęciem wyszukiwania. Jest to przydatne dla wpisów w pliku *hosts*, w których chcesz podawać same nazwy hosta, bez lokalnej domeny. Jeżeli podasz swoją domenę lokalną, zostanie ona usunięta przy poszukiwaniu hosta z dodaną nazwą domeny lokalnej, co pozwala na poprawne wyszukiwanie w pliku */etc/hosts*. Nazwa domeny, którą podajesz, musi kończyć się kropką (na przykład `linux.org.au.`), jeżeli `trim` ma działać poprawnie.

Opcje `trim` łączą się ze sobą. Możesz sprawić, że twój host będzie uznawany za lokalny w kilku domenach.

W przykładzie 6-1 pokazano plik *host.conf* dla hosta **vlager**.

Przykład 6-1. Przykładowy plik host.conf

```
# /etc/host.conf
# named dzia□a, ale nie mamy NIS-a (jeszcze)
order   bind,hosts
# Pozwalamy na wielokrotne adresy
multi   on
# Zabezpieczamy si□ przed próbami spoofingu
nospoof on
# obcinamy domen□ lokaln□ (nie jest to naprawd□ niezb□dne).
trim    vrew.com.
```

Zmienne środowiskowe resolvera

Ustawienia w pliku *host.conf* mogą być zmienione za pomocą szeregu zmiennych środowiskowych:

`RESOLV_HOST_CONF`

Ta zmienna określa, jaki plik ma być czytany zamiast */etc/host.conf*.

`RESOLV_SERV_ORDER`

Ta zmienna unieważnia opcję `order` zawartą w pliku *host.conf*. Usługi są podawane jako `hosts`, `bind` i `nis`, oddzielone spacjami, przecinkami, dwukropkami lub średnikami.

`RESOLV_SPOOF_CHECK`

Ta zmienna określa stopień ochrony przed spoofingiem. Podanie `off` wyłącza tę opcję. Wartości `warn` i `warn off` włączają sprawdzanie spoofingu, odpowiednio, przez włączenie i wyłącznie logowania. Wartość * włącza sprawdzanie spoofingu, ale pozostawia funkcję logowania zgodnie z tym, co jest zdefiniowane w pliku *host.conf*.

`RESOLV_MULTI`

Ta zmienna pozwala na podanie wartości `on` lub `off` i unieważnia opcję `multi` z pliku *host.conf*.

`RESOLV_OVERRIDE_TRIM_DOMAINS`
: Ta zmienna określa listę domen, które mają być obcinane, i unieważnia te podane w pliku *host.conf*. Obcinanie domen omówiliśmy wcześniej, przy opisie słowa kluczowego `trim`.

`RESOLV_ADD_TRIM_DOMAINS`
: Ta zmienna określa listę obcinanych domen, dodawaną do listy podanej w pliku *host.conf*.

Plik nsswitch.conf

Wersja 2. standardowej biblioteki GNU oferuje wydajniejszy i bardziej elastyczny mechanizm, który zastępuje starszy plik *host.conf*. Pojęcie usługi nazewniczej zostało rozszerzone tak, że obecnie jej plik zawiera wiele różnych informacji. Opcje konfiguracyjne dla różnych funkcji zadających zapytania do jej baz danych zostały z powrotem umieszczone w jednym pliku konfiguracyjnym o nazwie *nsswitch.conf*.

Plik *nsswitch.conf* pozwala administratorowi systemu skonfigurować szereg różnych baz danych. Ograniczymy omówienie do opcji związanych z rozwiązywaniem adresów IP hostów i sieci. Więcej informacji na temat innych funkcji możesz znaleźć w dokumentacji standardowej biblioteki GNU.

Opcje w pliku *nsswitch.conf* muszą występować w oddzielnych wierszach. Pola mogą być oddzielone białymi znakami (spacjami lub tabulatorami). Znak hasha (#) oznacza komentarz, który ciągnie się do następnego wiersza. Każdy wiersz opisuje określoną usługę – jedną z nich jest rozwiązywanie nazwy hosta. Pierwsze pole w każdym wierszu to nazwa bazy danych kończąca się dwukropkiem. Nazwa bazy związanej z rozwiązywaniem adresów hostów to `hosts`. Inna, związana z usługą nazewniczą baza to `networks`; służy do zamiany nazw sieci na ich adresy. Pozostała część każdego wiersza zawiera opcje określające sposób wyszukiwania w bazie danych.

Dostępne są następujące opcje:

`dns`
: Użycie systemu nazw domen (DNS) do rozwiązywania adresów. Ma to sens jedynie przy rozwiązywaniu adresów hostów, a nie sieci. Mechanizm ten wykorzystuje plik */etc/resolv.conf*, opisany w dalszej części tego rozdziału.

`files`
: Przeszukiwanie pliku lokalnego w poszukiwaniu nazwy hosta lub sieci i odpowiadających im adresów. Ta opcja wykorzystuje tradycyjne pliki */etc/hosts* i */etc/networks*.

`nis` lub `nisplus`
: Użycie systemu informacji sieciowej (NIS) do rozwiązywania adresów hostów lub sieci. NIS i NIS+ zostały szczegółowo omówione w rozdziale 13, *System informacji sieciowej*.

Kolejność, w jakiej są podane, decyduje o porządku zadawania zapytań o rozwiązanie nazwy. Lista kolejności zapytań znajduje się w opisie usługi umieszczonym

w pli ku */etc/nsswitch.conf*. Usługi są za py ty wa ne od le wej do pra wej. Do my śl nie poszu ki wa nie ko ńczy się, gdy roz wiąza nie na zwy się po wie dzie.

W przykładzie 6-2 po kaz uje my pro sty plik, na ślad ujący naszą kon fig ura cję wy korzy stującą starszą bi bliot ekę stan dard ową libc.

Przykład 6-2. Przykład owy plik nsswitch.conf

```
# /etc/nsswitch.conf
#
# Przyk□adowa konfiguracja funkcjonalno□ci GNU Name Service Switch.
# Informacje o tym pliku s□ dost□pne w pakiecie 'libc6-doc'.

hosts:          dns files
networks:       files
```

Za pis taki jak w po wyż sz ym przykładzie oznac za, że sys tem po szuk uje ho stów najpierw przez sys tem nazw do men, a na stępn ie, je żeli to się nie uda, w pli ku */etc/hosts*. Po szuk iwa nie nazw sie ci bę dzie re aliz owa ne tyl ko w oparc iu o plik */etc/networks*.

Możesz bardziej precyzyjnie sterować poszukiwaniami, jeśli skorzystasz z „elementów działania”. Pod po wia dają one ko lej ne kroki na pod staw ie wyn ików uzyska nych w po przedn iej pr óbie wy szuk iwa nia. Elem enty działania znaj dują się pomięd zy spe cyf ika cja mi usług i są otoc zone na wias ami kwa drat owy mi `[]`. Ogó lna skład nia dy rekt ywy działania jest na stęp ująca:

```
[ [!] status = dzia□anie ... ]
```

Ist nieją dwa mo żliwe działania:

`return`
: Po wo du je po wr ót do pro gra mu, któ ry pró bował roz wiązać na zwę. Je żeli pr óba wy szu ki wa nia się po wiodła, re solver zw róci szc zegółowe informacje, a w przeciw nym ra zie zw róci zero.

`continue`
: Re solver przej dzie do ko lej nej usługi na liś cie i spr óbuje za jej po mocą zna le źć nazwę.

Opcjo nal ny znak wy krzyk ni ka (!) mó wi, że sta tus po wi nien być od wrócony przed wy ko na niem te stu, czy li ozna cza „nie”.

Do puszc zalne war toś ci sta tusu, na któ rych możemy oper ować to:

`success`
: Żądany ad res zo stał zna le zio ny bez błędu. Do my śl ne działanie dla tego sta tu su to `return`.

`notfound`
: W wy szu ki wa niu nie wystąpił błąd, ale po szu ki wa ny host lub sieć nie mogą być zna le zio ne. Do my śl ne działanie dla tego sta tu su to `continue`.

`unavail`
: Usługa, do której zo stało za da ne za py ta nie, jest nie do stęp na. Może to ozna czać, że plik *hosts* lub *networks* jest nie czy tel ny dla usługi `files` lub że ser wer nazw

albo serwer NIS nie odpowiadają na usługę `dns` lub `nis`. Domyślne działanie dla tego statusu to `continue`.

`tryagain`
: Ten status mówi, że usługa jest tymczasowo niedostępna. W przypadku usługi `files` zwykle oznacza to, że dany plik jest zablokowany przez inny proces. W przypadku innych usług może to oznaczać tymczasową niemożność przyjęcia połączenia. Domyślne działanie dla tego statusu to `continue`.

Prostą ilustrację wykorzystania tego mechanizmu stanowi przykład 6-3.

Przykład 6-3. Przykładowy plik nsswitch.conf wykorzystujący dyrektywę działania

```
# /etc/nsswitch.conf
#
# Przyk□adowa konfiguracja funkcjonalno□ci GNU Name Service Switch.
# Informacje o tym pliku s□ dost□pne w pakiecie 'libc6-doc'.

hosts:          dns [!UNAVAIL=return] files
networks:       files
```

W tym przykładzie próbujemy znaleźć nazwę hosta za pomocą systemu usług nazewniczych domen. Jeżeli tylko zwrócony status nie oznacza niedostępności, resolver zwraca to, co znalazł. Jeżeli próba zapytania DNS zwróciła status niedostępności (wyłącznie w tym przypadku), resolver próbuje użyć lokalnego pliku */etc/hosts*. Oznacza to, że powinniśmy użyć pliku *hosts* tylko wtedy, gdy nasz serwer nazw z jakiegoś powodu jest niedostępny.

Konfigurowanie poszukiwania przez serwer nazw za pomocą pliku resolv.conf

Gdy konfigurujesz bibliotekę resolvera do korzystania z usługi nazewniczej BIND przy rozwiązywaniu nazw, musisz także wskazać serwery nazw, które mają być używane. Do tego celu służy oddzielny plik *resolv.conf*. Jeżeli plik ten nie istnieje lub jest pusty, resolver zakłada, że serwer nazw znajduje się na twoim hoście lokalnym.

Aby uruchomić serwer nazw na hoście lokalnym, musisz go oddzielnie skonfigurować, co wyjaśniamy w kolejnym podrozdziale. Jeżeli pracujesz w sieci lokalnej i masz możliwość wykorzystania istniejącego serwera nazw, nie omieszkaj tak zrobić. Jeżeli używasz komutowanego połączenia IP z Internetem, w pliku *resolv.conf* zwykle podajesz serwer nazw twojego dostawcy Internetu.

Najważniejszą opcją w pliku *resolv.conf* jest `name server`, zawierająca adres tego serwera nazw, który ma być używany. Jeżeli podasz kilka serwerów nazw, wpisując kilkukrotnie opcję `name server`, będą one sprawdzane w zadanej kolejności. W związku z tym najbardziej niezawodne serwery powinieneś umieszczać na początku. Bieżąca implementacja pozwala ci na umieszczenie w pliku *resolv.conf* trzech dyrektyw `name server`. Jeżeli nie zostanie podana opcja `name server`, resolver podejmie próbę połączenia z serwerem nazw na hoście lokalnym.

Dwie pozostałe opcje to `domain` i `search`, pozwalające na stosowanie skróconych nazw hostów w domenie lokalnej. Zwykle jeśli łączysz się za pomocą telnetu z in-

nym hostem w domenie lokalnej, nie musisz wpisywać pełnej nazwy. Wystarczy podać tylko nazwę krótką typu **gauss**. Resolver skojarzy ją z pozostałą częścią nazwy: **mathematics.groucho.edu**.

Tak właśnie działa dyrektywa `domain`. Pozwala podać domyślną nazwę domeny, która ma być dodawana, gdy DNS nie znajdzie nazwy hosta. Na przykład, gdy podamy nazwę **gauss**, resolver nie znajdzie jej w DNS-ie, ponieważ nie ma takiej domeny podstawowej. Gdy jako domenę domyślną wskażemy **mathematics.groucho.edu**, resolver powtórzy zapytanie o **gauss**. Tym razem wyszukiwanie się powiedzie.

Pewnie ci się to podoba, ale kiedy wyjdziesz poza domenę wydziału matematyki, musisz powrócić do pełnych nazw domen. Oczywiście chciałbyś mieć również skróty, takie jak **quark.physics** dla hostów z domeny wydziału fizyki.

Tu z pomocą przychodzi *lista przeszukiwania*. Można ją podać za pomocą opcji `search`, która jest uogólnieniem dyrektywy `domain`. Ta druga umożliwia wprowadzenie pojedynczej domeny domyślnej, natomiast ta pierwsza pozwala na podanie listy domen, które będą po kolei sprawdzane, aż poszukiwanie zakończy się powodzeniem. Elementy listy muszą być oddzielone spacjami lub tabulatorami.

Dyrektywy `search` i `domain` wzajemnie się wykluczają i nie mogą pojawić się w pliku więcej niż raz. Jeżeli zostanie podana któraś z opcji, resolver będzie próbował odgadnąć domyślną domenę na podstawie nazwy lokalnego hosta za pomocą wywołania systemowego *getdomainname(2)*. Jeżeli nazwa hosta lokalnego nie zawiera nazwy domeny, przyjmowana jest domena główna (ang. *root domain*).

Jeżeli zdecydujesz się umieścić dyrektywę `search` w pliku *resolv.conf*, powinieneś uważać na to, jakie domeny dodajesz do listy. Biblioteki resolvera wcześniejsze niż BIND 4.9 tworzyły domyślną listę przeszukiwania na podstawie nazwy domeny, jeżeli lista nie została podana. Domyślna lista składała się z samej domeny domyślnej oraz wszystkich jej domen nadrzędnych, aż do domeny głównej. Powodowało to pewne problemy, ponieważ zapytania DNS kończyły się na serwerach nazw, do których nigdy nie powinny dotrzeć.

Załóżmy, że jesteś w browarze wirtualnym i chcesz zalogować się do **foot.groucho.edu**. Przypuśćmy, że omsknął ci się palec i wpisałeś **foo** zamiast **foot**, czy li podałeś nazwę hosta, która nie istnieje. Serwer nazw GMU powie ci, że nie zna takiego hosta. W przypadku listy poszukiwań starego typu, resolver próbowałby dołączać do nazwy hosta nazwy **vbrew.com** i **com**. Ta ostatnia nazwa może być problematyczna, ponieważ domena **groucho.edu.com** może istnieć naprawdę. Co więcej, być może w tej domenie serwer znajdzie jakiegoś hosta **foo** i wskaże nie na to, co potrzeba.

Takie fałszywe wyniki poszukiwań mogą zagrażać systemowi bezpieczeństwa niektórych aplikacji. Dlatego zwykle powinieneś zawężać domeny na twojej liście poszukiwań do swojej lokalnej organizacji lub czegoś w tym rodzaju. Na wydziale matematyki uniwersytetu Groucho Marx lista poszukiwań powinna być skonfigurowana na **maths.groucho.edu** i **groucho.edu**.

Jeżeli zrozumienie domyślnych domen sprawia ci kłopot, przyjrzyj się poniższemu przykładowi pliku *resolv.conf* dla wirtualnego browaru:

```
# /etc/resolv.conf
# Nasza domena
domain       vbrew.com
#
# Jako g□ównego serwera nazw u□ywamy vlager
name server 172.16.1.1
```

Przy rozwiązywaniu nazwy **vale**, resolverowi nie uda się znaleźć **vale**, ale znajdzie **vale.vbrew.com**.

Siła resolvera

Jeżeli obsługujesz sieć lokalną w obrębie dużej sieci, zdecydowanie powinieneś korzystać z głównych serwerów nazw, jeżeli takie są dostępne. Serwery nazw mają obszerną pamięć podręczną, która przyspiesza odpowiedzi na powtórne zapytania, a wszystkie zapytania są kierowane właśnie do tych serwerów. Jednak ten model ma jedną wadę: gdyby ogień zniszczył kabel szkieletu na uniwersytecie Olafa, nie można byłoby pracować w wydziałowej sieci LAN, ponieważ resolver nie mógłby się skomunikować z żadnym z serwerów nazw. Taka sytuacja powoduje trudności z większością usług sieciowych, takich jak logowanie się z X terminali czy drukowanie.

Choć nie zbyt często zdarza się, że sieciszkielet owe campusu płoną, to warto zabezpieczyć się przed takim wypadkiem.

Jednym z rozwiązań jest skonfigurowanie lokalnego serwera nazw, który rozwiązuje nazwy z domeny lokalnej i przekazuje wszystkie zapytania o inne hosty do głównych serwerów. Oczywiście ma to sens jedynie wtedy, gdy posiadasz własną domenę.

Alternatywą może być utrzymywanie zapasowej listy hostów dla twojej domeny czy sieci lokalnej w pliku */etc/hosts*. Można to zrobić bardzo łatwo. Po prostu konfigurujemy bibliotekę resolvera tak, aby w pierwszej kolejności zadawała zapytania do DNS-u, a następnie sprawdzała plik hostów. W pliku */etc/host.conf* powinieneś wpisać: **`order bind hosts`**, a w pliku */etc/nsswitch.conf*: **`hosts: dns files`**. W ten sposób resolver wykorzysta plik hostów, jeżeli główny serwer nazw będzie nieosiągalny.

Jak działa DNS

DNS porządkuje nazwy hostów w hierarchii domen. *Domena* to zbiór ośrodków maszyn, które są jakoś powiązane ze sobą. Na przykład tworzą sieć (tak jak wszystkie maszyny w campusie lub wszystkie hosty sieci BITNET), lub należą do pewnej organizacji (np. rządu Stanów Zjednoczonych), lub leżą blisko siebie. Na przykład uniwersytety są zwykle grupowane w domenie **edu**, a każdy uniwersytet czy college używa oddzielnej *poddomeny*, w której są zebrane jego hosty. Uniwersytet Groucho Marx posiada domenę **groucho.edu**, natomiast sieć lokalna wydziału matematyki

znajduje się w poddomenie **maths.groucho.edu**. W nazwach hostów z sieci wydziału powinny znajdować się domeny, a więc **erdos** będzie znany jako **erdos.maths.groucho.edu**. Taka nazwa to *pełna nazwa domenowa* zwykle identyfikująca dany host w skali świata.

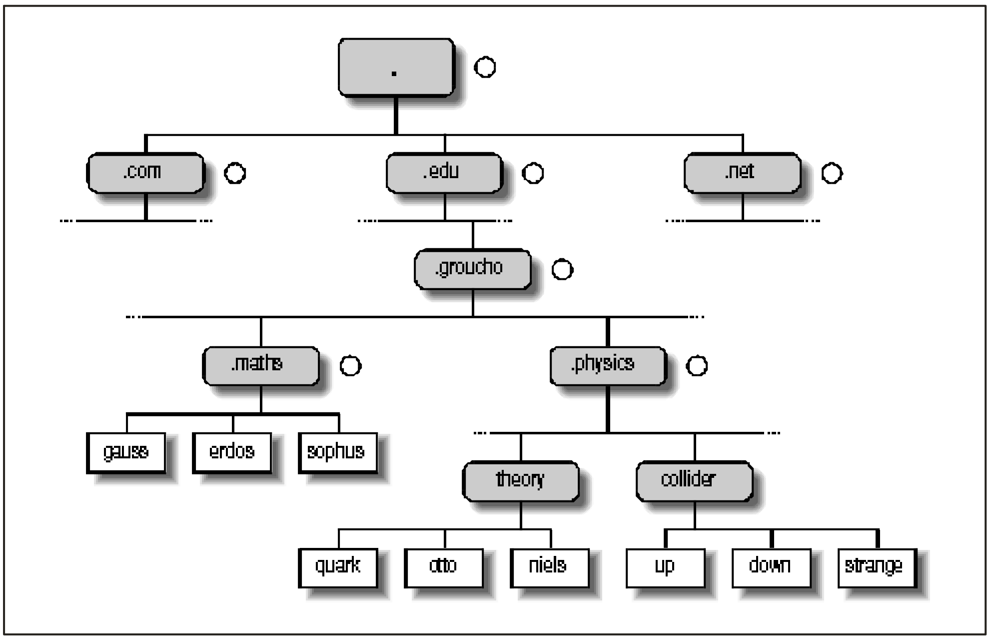
Rysunek 6-1 pokazuje podział przestrzeni nazw. Wpis u góry drzewa, po prostu kropka, jest nazywany *domeną główną* (ang. *root domain*) i obejmuje wszystkie pozostałe domeny. Aby pokazać, że nazwa hosta jest pełną nazwą domenową, a nie nazwą względną dla jakiejś (ukrytej) domeny lokalnej, czasem jest ona pisana z kropką na końcu. Kropka ta oznacza, że ostatnim członem nazwy jest domena główna.

W zależności od położenia nazwy w hierarchii, domena może być nazywana domeną najwyższego poziomu, drugiego poziomu lub trzeciego poziomu. Poziomów jest jeszcze więcej, ale rzadko są używane. Poniższa lista pokazuje kilka domen najwyższego poziomu, z którymi często się możesz spotkać:

Domena	***Opis***
edu	(Głównie w USA) Instytucje edukacyjne, na przykład uniwersytety.
com	Organizacje i firmy komercyjne.
org	Organizacje niekomercyjne. Prywatne sieci UUCP często należą do tej domeny.
net	Gateway'e i inne hosty administracyjne w sieci.
mil	Amerykańskie instytucje wojskowe.
gov	Amerykańskie instytucje rządowe.
uucp	Oficjalnie, wszystkie nazwy wcześniej używane jako nazwy UUCP bez domen zostały przeniesione do tej domeny.

Historycznie pierwsze cztery domeny były przypisane Stanom Zjednoczonym, ale ostatnio w praktyce na zewnętrznej kładzie się nacisk na globalny charakter tych domen, również w znaczeniu terytorialnym, bo przecież są domenami globalnymi najwyższego rzędu (ang. *global Top Level Domains* – gTLD). Prowadzone są negocjacje na temat rozszerzenia zakresu gTLD, co może zaowocuje większymi możliwościami wyboru w przyszłości.

Poza Stanami Zjednoczonymi, każdy kraj używa domeny najwyższego poziomu w postaci własnego dwuliterowego kodu kraju zdefiniowanego w normie ISO-3166. Na przykład Finlandia używa domeny **fi**; **fr** to domena dla Francji, **de** dla Niemiec, zaś **au** dla Australii. Na pozostałych poziomach hierarchii (tych poniżej domeny najwyższego poziomu), organizacja NIC każdego kraju może porządkować nazwy hostów dowolnie. Australia posiada domeny drugiego poziomu podobne do międzynarodowych domen najwyższego poziomu, czyli **com.au** i **edu.au**. Inne kraje, na przykład Niemcy, nie wykorzystują tego dodatkowego poziomu, a raczej wydłużają nazwę od noszącą się bezpośrednio do firmy, z którą jest związana dana domena. Nie jest niczym niezwykłym spotkanie domeny **ftp.informatik.uni-enlargen.de**. Ale to już sprawa Niemców.



Rysunek 6-1. *Fragment przestrzeni nazw domen*

Oczywiście takie domeny narodowe nie oznaczają, że host w danej domenie znajduje się fizycznie w danym kraju – oznacza to jedynie, że host został zarejestrowany w organizacji NIC danego kraju. Na przykład załóżmy, że szwedzki przemysłowiec ma filię swojej firmy w Australii, ale wszystkie hosty pracujące w tej filii mogą być zarejestrowane w domenie najwyższego poziomu **se**.

Uporządkowanie przestrzeni nazw w hierarchii nazw domen to eleganckie rozwiązanie problemu niepowtarzalności nazw. W DNS-ie wystarczy, że nazwa hosta będzie unikatowa we własnej domenie, a pozostanie taka na całym świecie. Co więcej, pełne nazwy domenowe są łatwe do zapamiętania. Już te kilka faktów przemawia za podziałem dużej domeny na kilka domen podrzędnych.

DNS daje ci jeszcze więcej możliwości. Pozwala także na przekazywanie władzy nad poddomenami ich administratorom. Na przykład osoby obsługujące centrum komputerowe uniwersytetu Groucho Max mogą stworzyć poddomenę dla każdego wydziału. Już spotkaliśmy poddomeny **math** i **physics**. Kiedy stwierdzą, że sieć na wydziale fizyki jest zbyt duża i trudno nią zarządzać z zewnątrz (w końcu wiadomo, że fizycy to niesforna grupa ludzi), mogą po prostu przekazać kontrolę nad domeną **physics.groucho.edu** administratorom tej sieci. Administratorzy będą mieli prawo nazywać hosty, jak chcą, i przypisywać adresy IP z ich sieci w dowolnie wybrany przez siebie sposób, bez konieczności uwzględniania jakichkolwiek sugestii z zewnątrz.

W tym celu przestrzeń nazw jest podzielona na *strefy* (ang. *zones*) oparte na domenach. Zwróć uwagę na subtelną różnicę pomiędzy *strefą* a *domeną*: domena **groucho.edu** zawiera wszystkie hosty z uniwersytetu Groucho Marx, natomiast strefa **groucho.edu** zawiera jedynie hosty zarządzane bezpośrednio przez centrum komputerowe – na przykład te na wydziale matematyki. Hosty na wydziale fizyki należą do innej strefy, a mianowicie **physics.groucho.edu**. Na rysunku 6-1 początek strefy jest zaznaczony małym kółkiem przy nazwie domeny.

Poszukiwanie nazw w DNS-ie

Na pierwszy rzut oka wydaje się, że cały ten podział na domeny i strefy bardzo komplikuje rozwiązywanie nazw. W zasadzie, jeżeli żad na władza centralna nie kontroluje przypisywania nazw hostom, to skąd ma je znać skromna aplikacja?

Teraz przejdźmy do naprawdę pomysłowej części DNS-u. Gdybyś chciał znaleźć adres IP hosta **erdos**, DNS powiedziałby: „Idź, zapytaj ludzi, którzy go obsługują, a oni ci powiedzą".

W rzeczywistości DNS to gigantyczna rozproszona baza danych. Jest zaimplementowana w postaci tak zwanych serwerów nazw, które dostarczają informacji o zadanej domenie lub zestawie domen. Każda strefa ma przynajmniej dwa (lub kilka) serwerów nazw, które są źródłem wszelkich informacji o hostach z tej strefy. Aby uzyskać adres IP **erdosa**, wystarczy skontaktować się z serwerem nazw w strefie **groucho.edu**, a on zwróci ci wymagane dane.

Możesz pomyśleć: łatwo się mówi, ale trudniej to zrobić. Skąd mam wiedzieć, jak znaleźć serwer nazw uniwersytetu Groucho? Jeżeli twój komputer nie jest wyposażony w wyrocznię, znajdującą adresy, może za nią posłużyć także DNS. Gdy twoja aplikacja chce znaleźć informacje o **erdosie**, kontaktuje się z lokalnym serwerem nazw, który na jej rzecz wykonuje tak zwane zapytania iteracyjne. Rozpoczyna od wysyłania zapytania o adres **erdos.maths.groucho.edu** do serwera nazw domeny głównej. Serwer nazw domeny głównej rozpoznaje, że ta nazwa nie należy do strefy będącej w jego władzy, ale należy do domeny **edu**. Odpowiada naszemu serwerowi nazw, żeby w celu uzyskania dokładniejszych informacji, skontaktował się z serwerem nazw strefy **edu**, i wysyła listę wszystkich serwerów nazw **edu** wraz z ich adresami. Twój lokalny serwer nazw działa dalej i wysyła zapytanie do jednego z polecanych serwerów, na przykład **a.isi.edu**. Podobnie jak serwer nazw domeny głównej, tak i **a.isi.edu** wie, że ludzie z **grouche.edu** mają własną strefę i wskazuje ich serwery. Lokalny serwer nazw następnie kieruje zapytanie o **erdosa** do jednego z nich, który z kolei ostatecznie identyfikuje nazwę jako należącą do jego strefy i zwraca odpowiadający jej adres IP.

Wygląda na to, że aby znaleźć jeden marny adres IP, trzeba wygenerować spory ruch, ale to i tak nic w porównaniu z liczbą danych, jaka musiałaby być przesyłana, gdybyśmy wciąż korzystali z pliku *HOSTS.TXT*. Schemat ten jednak daje się udoskonalić.

Aby skrócić czas odpowiedzi na przyszłe zapytania, serwer nazw zapisuje uzyskane informacje w lokalnej *pamięci podręcznej*. Tak więc, jeżeli ktoś z twojej sieci lokalnej

będzie chciał znowu znaleźć adres hosta w domenie **groucho.edu**, twój serwer nazw skontaktuje się bezpośrednio z serwerem nazw w tej domenie*.

Oczywiście serwer nazw nie będzie przechowywał tej informacji wiecznie. Po jakimś czasie ją usunie. Czas jej przechowywania jest nazywany *czasem życia* (ang. *time to live*), w skrócie TTL. Wszystkim danym w DNS-ie administrator danej strefy przypisuje TTL.

Typy serwerów nazw

Serwery nazw, które przechowują wszystkie informacje o hostach z danej strefy, są nazywane *autorytatywnymi* (ang. *authoritative servers*) dla tej strefy, a czasami *głównymi serwerami nazw* (ang. *master name servers*). Wszelkie zapytania o hosta w danej strefie docierają w końcu do serwerów głównych.

Serwery główne muszą być doskonale zsynchronizowane. Tak więc administrator strefy musi stworzyć jeden serwer *podstawowy* (ang. *primary*), który ładuje informacje o strefie z plików z danymi, i serwery *zapasowe* (ang. *secondary*), które przesyłają dane o strefie z serwera podstawowego w równych odstępach czasu.

Dobrze jest mieć kilka serwerów nazw, gdyż można równomiernie rozłożyć obciążenie i zagwarantować lepszą niezawodność. Gdy jedna z maszyn, na której działa serwer nazw w łagodny sposób przestanie działać, na przykład system operacyjny ulegnie awarii lub straci połączenie z siecią, wszystkie zapytania będą kierowane do innych serwerów. Oczywiście taki schemat nie zabezpiecza przed pomyłkami (wynikającymi z błędów w oprogramowaniu lub w samym programie serwera), które powodują błędne odpowiedzi na wszystkie zapytania DNS.

Możesz także uruchomić serwer nazw, który nie jest autorytatywny dla danej domeny**. Jest to przydatne, gdyż taki serwer będzie w stanie realizować zapytania DNS dla aplikacji działających w sieci lokalnej i zatrzyma informacje w pamięci podręcznej. Stąd serwery takie są nazywane *serwerami pamięci podręcznej* (ang. *caching-only servers*).

Baza danych DNS

Widzieliśmy, że DNS nie tylko podaje adresy IP hostów, ale także wymienia informacje na temat serwerów nazw. Bazy danych DNS mogą mieć w praktyce wiele różnych typów wpisów.

Pojedyncza porcja informacji z bazy DNS nazywa się *rekordem zasobu* (ang. *resource record* - RR). Każdy rekord przynależy do jakiegoś typu i klasy rekordów. Typ opisuje rodzaj danych reprezentowanych w rekordzie. Natomiast klasa określa rodzaj sieci, jakiej dotyczy. Klasa służy do obsługi różnych schematów adresowania, jak adresy

---

* Jeżeli informacje nie byłyby gromadzone w pamięci podręcznej, DNS byłby równie nieefektywny jak inne metody, ponieważ każde zapytanie wymagałoby skontaktowania się z serwerami nazw domeny głównej.

** Serwer nazw musi zapewnić przynajmniej usługę nazewniczą dla **localhost** i odwrotne wyszukiwanie dla **127.0.0.1**).

IP (klasa IN), adresy Hesiod (używane przez system Kerberos MIT) i kilka innych. Prototypowym rekordem zasobu jest rekord A wiążący pełną nazwę domenową z adresem IP.

Host może być znany pod więcej niż jedną nazwą. Na przykład możesz mieć komputer, który posiada serwery FTP i WWW dostępne pod dwoma nazwami: **ftp.machine.org** i **www.machine.org**. Jednak jedna z tych nazw musi być oficjalną lub *kanoniczną* nazwą hosta, natomiast pozostałe są po prostu jej aliasami. Różnica polega na tym, że kanoniczna nazwa hosta jest związana z rekordem A, natomiast pozostałe mają jedynie rekord typu CNAME, wskazujący na nazwę kanoniczną.

Nie będziemy tu przedstawiać wszystkich typów rekordów, ale podamy krótki przykład. Przykład 6-4 pokazuje część bazy danych domeny, która jest ładowana do serwerów nazw dla strefy **physics.groucho.edu**.

Przykład 6-4. Fragment pliku named.hosts wydziału fizyki

```
; Authoritative Information on physics.groucho.edu.
@  IN SOA niels.physics.groucho.edu. janet.niels.physics.groucho.edu {
                 1999090200        ; numer wersji
                 360000            ; od□wie□anie
                 3600              ; ponowna próba
                 3600000           ; wyga□ni□cie
                 3600              ; domy□lny ttl
                }
;
; serwery nazw
              IN   NS   niels
              IN   NS   gauss.maths.groucho.edu.
gauss.maths.groucho.edu. IN A 149.76.4.23
;
; fizyka teoretyczna (podsie□ 12)
niels         IN   A   149.76.12.1
              IN   A   149.76.1.12
name server   IN CNAME   niels
otto          IN   A   149.76.12.2
quark         IN   A   149.76.12.4
down          IN   A   149.76.12.5
strange       IN   A   149.76.12.6
...
; Laboratorium Collider (podsie□ 14)
boson         IN   A   149.76.14.1
muon          IN   A   149.76.14.7
bogon         IN   A   149.76.14.12
...
```

Poza rekordami A i CNAME, możesz zobaczyć na początku pliku specjalny rekord zajmujący kilka wierszy. Jest to rekord zasobów SOA (lub krócej: rekord SOA); SOA to skrót od angielskiego *start of authority* – początek władzy). Rekord ten zawiera ogólne informacje o strefie, dla której serwer jest autorytatywny i składa się między innymi z domyślnego czasu życia odnoszące go się do wszystkich rekordów.

Zauważ, że wszystkie nazwy w pliku przykładowym, które nie kończą się kropką, powinny być interpretowane względem domeny **physics.groucho.edu**. Nazwa specjalna (@) użyta w przykładowym rekordzie SOA odnosi się do samej nazwy domeny.

Wcześniej zauważyliśmy, że serwery nazw domeny **groucho.edu** skądś wiedzą o strefie **physics**, tak że mogą zadawać pytania do jej serwerów nazw. Zwykle robi się to za pomocą pary rekordów: rekordu NS, który podaje pełną nazwę domenową serwera, i rekordu A, który wiąże adres z tą nazwą. Ponieważ te rekordy wiążą przestrzeń nazw, często są nazywane *rekordami klejącymi* (ang. *glue records*). Są to jedyne rekordy, w których strefa nadrzędna w rzeczywistości przechowuje informacje o hostach strefy podrzędnej. Rekordy klejące wskazują na serwery nazw domeny **physics.groucho.edu**, jak pokazano w przykładzie 6-5.

Przykład 6-5. Fragment pliku named.hosts z GMU

```
; Dane dla strefy groucho.edu
@  IN  SOA vax12.gcc.groucho.edu. joe.vax12.gcc.groucho.edu. {
                      1999070100       ; numer wersji
                      360000           ; od□wie□anie
                      3600             ; ponowna próba
                      3600000          ; wyga□ni□cie
                      3600             ; domy□lny ttl
               }
...
;
;rekordy klej□ce dla strefy physics.groucho.edu
physics          IN       NS      niels.physics.groucho.edu.
                 IN       NS      gauss.maths.groucho.edu.
niels.physics    IN       A       149.76.12.1
gauss.maths      IN       A       149.76.4.23
...
```

Wyszukiwanie odwrotne

Znajdowanie adresu IP należącego do hosta jest pewnie najpowszechniejszym zastosowaniem systemu nazw domen, ale czasem chcesz znaleźć kanoniczną nazwę hosta odpowiadającą adresowi. Znajdowanie nazwy hosta jest nazywane *odwzorowaniem odwrotnym* (ang. *reverse mapping*) i jest używane przez pewne usługi sieciowe do weryfikacji tożsamości klienta. Wyszukiwanie odwrotne, przeprowadzane w oparciu o plik *hosts*, polega po prostu na przeszukaniu pliku i znalezieniu w nim hosta, do którego należy poszukiwany adres IP. W przypadku DNS-u skrupulatne przeszukiwanie przestrzeni nazw jest wykluczone. Zamiast tego została stworzona specjalna domena **in-addr.arpa**, która zawiera adresy IP wszystkich hostów zapisane w odwrotnej notacji kropkowej. Na przykład adres IP **149.76.12.4** odpowiada nazwie **4.12.76.149.in-addr.arpa**. Rekord zasobu, który łączą nazwy z odpowiadającymi im kanonicznymi nazwami hostów, jest typu PTR.

Tworzenie zarządzanej przez nas strefy zwykle oznacza, że jej administratorzy w pełni kontrolują sposób przypisywania adresów do nazw. Ponieważ zwykle mają w swoich rękach jedną lub więcej sieci lub podsieci IP, odwzorowanie stref DNS na sieci IP jest typu jedna-na-wiele. Na przykład wydział fizyki zawiera podsieci **149.76.8.0**, **149.76.12.0** i **149.76.14.0**.

Oznacza to, że wraz ze strefą **physics** w domenie **in-addr.arpa** muszą być stworzone i przekazane administratorom sieci na wydziale nowe strefy: **8.76.149.in-addr.arpa**, **12.76.149.in-addr.arpa** i **14.76.149.in-addr.arpa**. W przeciwnym razie dodanie nowego hosta w laboratorium Collider wymagałoby skontaktowania się z domeną nadrzędną i wprowadzenia nowego adresu do pliku strefy **in-addr.arpa**.

Baza danych strefy dla podsieci 12 została pokazana w przykładzie 6-6. Odpowiednie rekordy klejące w bazie danych strefy nadrzędnej zostały pokazane w przykładzie 6-7.

Przykład 6-6. Fragment z pliku named.rev dla podsieci 12

```
; domena 12.76.149.in-addr.arpa
@  IN SOA  niels.physics.groucho.edu. janet.niels.physics.groucho.edu. {
                 1999090200 360000 3600 3600000 3600
            }
2      IN PTR          otto.physics.groucho.edu.
4      IN PTR          quark.physics.groucho.edu.
5      IN PTR          down.physics.groucho.edu.
6      IN PTR          strange.physics.groucho.edu.
```

Przykład 6-7. Fragment pliku named.rev dla sieci 149.76

```
; domena 76.149.in-addr.arpa
@  IN SOA  vax12.gcc.groucho.edu. joe.vax12.gcc.groucho.edu. {
                    1999070100 360000 3600 3600000 3600
                 }
...
; posie□ 4: wydzia□ matematyki
1.4         IN   PTR         sophus.maths.groucho.edu.
17.4        IN   PTR         erdos.maths.groucho.edu.
23.4        IN   PTR         gauss.maths.groucho.edu.
...
; podsie□ 12: wydzia□ fizyki, oddzielna strefa
12          IN   NS          niels.physics.groucho.edu.
            IN   NS          gauss.maths.groucho.edu.
niels.physics.groucho.edu.   IN A 149.76.12.1
gauss.maths.groucho.edu.     IN A 149.76.4.23
...
```

Strefy systemu **in-addr.arpa** mogą być tworzone tylko jako nadzbiory sieci IP. Jeszcze poważniejszym ograniczeniem jest to, że maski tych sieci muszą przestrzegać granic bajtowych*. Wszystkie podsieci uniwersytetu Groucho Marx mają maskę sieci **255.255.255.0**, a więc strefa **in-addr.arpa** może być utworzona dla każdej podsieci. Jednak, gdyby maska miała postać **255.255.255.128**, utworzenie stref dla podsieci **149.76.12.128** byłoby niemożliwe, ponieważ nie ma sposobu na poinformowanie DNS-u, że domena **12.76.149.in-addr.arpa** została podzielona na dwie strefy, gdzie hosty mają numery odpowiednio z zakresów: od **1** do **127** i od **128** do **255**.

* Ograniczenie to nie dotyczy najnowszej wersji BIND 8 (-przyp. tłum.).

Eksploatacja named

named (wymawiaj: *nejm-di*) umożliwia korzystanie z usługi DNS na większości komputerów uniksowych. Jest to program serwera, oryginalnie stworzony dla BSD, który służy do udostępniania usług na zewnicznych klientom oraz innym serwerom nazw. Przez jakiś czas był używany BIND w wersji 4, obecny w wielu dystrybucjach Linuksa. Nowa wersja, o numerze 8 została wprowadzona w większości dystrybucji Linuksa i znacznie różni się od poprzednich wersji*. Ma wiele nowych funkcji, takich jak dynamiczne uaktualnianie DNS-u, powiadomienie o zmianach w DNS-ie, lepsza wydajność i nowa składania pliku konfiguracyjnego. Szczegóły znajdziesz w dokumentacji załączonej do pakietu dystrybucyjnego.

Ten podrozdział wymaga rozumienia działania DNS-u. Jeżeli czujesz się jak na tureckim kazaniu, może warto ponownie sięgnąć do poprzedniego podrozdziału *Jak działa DNS*.

named zwykle jest uruchamiany w czasie startu systemu i działa aż do zatrzymania maszyny. Implementacje wcześniejszych wersji BIND pobierały swoje informacje z pliku konfiguracyjnego */etc/named.boot* i różnych plików zawierających odwzorowania nazw domen na adresy. Te ostatnie są nazywane *plikami stref*. Wersje BIND od wersji 8 wzwyż wykorzystują natomiast plik o nazwie */etc/named.conf*.

Aby uruchomić *named*, wprowadź:

```
# /usr/sbin/named
```

named uruchomi się i odczyta plik *named.boot* oraz pliki stref w nim wskazane. Zapisze ID swojego procesu w postaci pliku ASCII o nazwie */var/run/named.pid*, ściągnie wszelkie pliki stref z serwerów podstawowych, o ile będzie taka potrzeba, i zacznie nasłuchiwać na porcie 53, oczekując na zapytania DNS.

Plik named.boot

Plik konfiguracyjny wcześniejszej wersji BIND miał bardzo prostą strukturę. Natomiast ten plik dla wersji 8. ma znacznie trudniejszą składnię, obsługującą wiele nowo wprowadzonych funkcji. Nazwa tego pliku zmieniła się z */etc/named.boot* na */etc/named.conf*. Skupimy się na konfigurowaniu starszej wersji, ponieważ większość dystrybucji prawdopodobnie wciąż jeszcze jej używa, ale pokażemy także równoważny plik *named.conf*, żeby zilustrować różnice i powiemy, jak konwertować stary format na nowy.

Plik *named.boot* jest generalnie niewielki i zawiera jedynie kilka wskazań do plików głównych z informacjami o strefie i do innych serwerów. Komentarze rozpoczynają się hashem (#) lub średnikiem (;) i ciągną się do następnego wiersza. Za nim bardziej szczegółowo omówimy format *named.boot*, przyjrzymy się przykładowemu plikowi z hosta **vlager** pokazanemu w przykładzie 6-8.

* BIND 4.9 został stworzony przez Paula Vixie, *paul@vix.com*, ale obecnie BIND jest utrzymywany przez Internet Software Consortium: *bind-bugs@isc.org*.

Przykład 6-8. Plik named.boot dla hosta vlager

```
;
; plik /etc/named.boot dla hosta vlager.vbrew.com
;
directory    /var/named
;
;            domena                   plik
;--------------------
cache        .                        named.ca
primary      vbrew.com                named.hosts
primary      0.0.127.in-addr.arpa     named.local
primary      16.172.in-addr.arpa      named.rev
```

Przyjrzyjmy się kolejno każdej dyrektywie. Słowo kluczowe `directory` informuje program *named*, że wszystkie dalej wymienione pliki, czyli w tym przykładzie pliki stref, znajdują się w katalogu */var/named*. W rezultacie jest nieco mniej pisania.

Słowo kluczowe `primary` widoczne w tym przykładzie ładuje informacje do *named*. Informacje te są brane z plików głównych podanych jako ostatnie parametry w wierszu. Te pliki zawierają rekordy zasobów DNS, którym przyjrzymy się dalej.

W tym przykładzie skonfigurowaliśmy *named* jako podstawowy serwer nazw dla trzech domen, co pokazują trzy dyrektywy `primary`. Pierwsza z nich nakazuje programowi *named* działać jako podstawowy serwer dla domeny **vbrew.com** i odczytywać dane o strefie z pliku *named.hosts*.

Słowo kluczowe `cache` ma szczególne znaczenie i powinno być obecne na wszystkich komputerach, na których działa serwer nazw. Powoduje ono, że *named* włącza swoją pamięć podręczną i ładuje *wskazania do serwera nazw domeny głównej* (ang. *root name server hints*) z pliku pamięci podręcznej (w naszym przykładzie *named.ca*). Wrócimy do tego w poniższej liście.

Oto lista najważniejszych opcji, jakich możesz używać w pliku *named.boot*:

`directory`
: Ta opcja wyznacza katalog, w którym znajdują się pliki stref. Nazwy plików w innych opcjach mogą być podane względem tego katalogu. Wielokrotnie używając dyrektywy `directory`, można określić kilka katalogów. Standard systemu plików Linuksa mówi, że powinno się używać katalogu */var/named*.

`primary`
: Ta opcja przyjmuje jako argument nazwę domeny i nazwę pliku oraz mówi, że lokalny serwer jest autorytatywny dla danej domeny. Jako serwer podstawowy, *named* ładuje informacje o strefie z zadanego pliku głównego.

  W każdym pliku boot będzie istniał przynajmniej jeden wpis `primary` dotyczący odwrotnego odwzorowania sieci **127.0.0.0** – lokalnej sieci pętli zwrotnej.

`secondary`
: Ta dyrektywa jest używana z nazwą domeny, listą adresów i nazwą pliku jako argumentami. Mówi, że lokalny serwer jest serwerem zapasowym (ang. *secondary master server*) dla danej domeny.

  Serwer zapasowy także zawiera autorytatywne dane o domenie, ale nie odczytuje ich z plików, tylko próbuje je ładować z serwera podstawowego. W liście ad-

resów należy podać *named* przynajmniej jeden adres IP serwera podstawowego. Serwer lokalny kontaktuje się kolejno z każdym z nich, aż uda mu się poprawnie skopiować bazę danych stref, która następnie jest zapisywana w pliku zapasowym o nazwie podanej jako trzeci argument. Jeżeli żaden z serwerów głównych nie odpowiada, dane o strefie są odczytywane z pliku zapasowego.

named próbuje odświeżać dane o strefie w regularnych odstępach czasu. Proces ten wyjaśniamy dalej w połączeniu z rekordem zasobu SOA.

`cache`
: Ta opcja wymaga podania nazwy domeny i nazwy pliku jako argumentów. Plik zawiera wskazania do serwerów nazw domeny głównej, czyli listę rekordów z ich nazwami. Rozpoznawane są jedynie rekordy NS i A. `domain` powinno być ustawione na nazwę domeny głównej, czyli po prostu kropkę (.).

  Ta informacja jest kluczowa dla *named*. Jeżeli w pliku startowym nie występuje dyrektywa `cache`, *named* nie utworzy w ogóle lokalnej pamięci podręcznej. Taka sytuacja (brak tej funkcji) poważnie zmniejszy wydajność i zwiększy obciążenie sieci, jeżeli przepytywane serwery nie znajdują się w sieci lokalnej. Co więcej, *named* nie będzie w stanie skontaktować się z żadnym z serwerów nazw domeny głównej, a więc nie będzie mógł rozwiązać adresów innych, niż te, dla których jest autorytatywny. Wyjątkiem od tej reguły są serwery przekazujące (ang. *forwarding servers*; zobacz opcja `forwarders` opisana poniżej).

`forwarders`
: Ta dyrektywa wymaga jako argumentu listy adresów z separatorami w postaci białych znaków. Adresy IP na tej liście odpowiadają serwerom nazw, które *named* może pytać, jeżeli nie uda mu się odpowiedzieć na zapytanie na podstawie lokalnej pamięci podręcznej. Są one sprawdzane po kolei, aż któryś odpowie na zapytanie. Zwykle w tym miejscu podajesz serwer nazw swojego dostawcy sieci lub inny dobrze znany serwer.

`slave`
: Ta dyrektywa powoduje, że serwer nazw jest definiowany jako serwer *podległy* (ang. *slave server*). Nigdy sam nie realizuje zapytań rekurencyjnych, a jedynie przekazuje je do serwerów określonych w dyrektywie `forwarders`.

Istnieją dwie opcje, których tutaj nie opisujemy: `sortlist` i `domain`. W plikach baz danych można używać także dwóch innych dyrektyw: `$INCLUDE` i `$ORIGIN`. Ponieważ są one rzadko potrzebne, nie opisujemy ich tutaj.

Plik host.conf dla wersji BIND 8

BIND w wersji 8 wprowadza szereg nowych funkcji i wraz z nimi nową składnię pliku konfiguracyjnego. Plik *named.boot* ze swoimi prostymi, jednowierszowymi dyrektywami został zastąpiony przez plik *named.conf*, który ma składnię podobną do *gated*, a więc przypominającą składnię pliku źródłowego w języku C.

Nowa składnia jest bardziej skomplikowana, ale na szczęście przygotowano narzędzie, które automatycznie konwertuje starą składnię na nową. W pakiecie źródłowym BIND 8 dodano program *named-bootconf.pl* napisany w Perlu, który odczytu-

je istniejący plik *named.boot* z `stdin` i konwertuje go na równoważny plik w formacie *named.conf*, wypisywany na `stdout`. Aby użyć konwertera, musisz mieć zainstalowany interpreter Perla.

Skryptu używa się w następujący sposób:

```
# cd /etc
# named-bootconf.pl <named.boot >named.conf
```

Tworzy on plik *named.conf*, podobny do pokazanego w przykładzie 6-9. Usunęliśmy kilka pomocnych komentarzy, jakie dodaje skrypt, aby lepiej było widać prawie bezpośredni związek pomiędzy starą i nową składnią.

Przykład 6-9. Równoważny plik named.conf z serwera vlager dla wersji 8 BIND-a

```
//
// plik /etc/named.boot dla serwera vlager.vbrew.com
options {
        directory "/var/named";
};

zone "." {
        type hint;
        file "named.ca";
};

zone "vbrew.com" {
        type master;
        file "named.hosts";
};

zone "0.0.127.in-addr.arpa" {
        type master;
        file "named.local";
};

zone "16.172.in-addr.arpa" {
        type master;
        file "named.rev";
};
```

Jeżeli przyjrzysz się uważniej, spostrzeżesz, że każda z jednowierszowych dyrektyw pliku *named.boot* została zamieniona w pliku *named.conf* na dyrektywę w stylu języka C, ujętą w nawiasy {}.

Komentarze, które w pliku *named.boot* były oznaczone średnikiem (;), tutaj są sygnalizowane dwoma ukośnikami (//).

Dyrektywa `directory` została zamieniona na akapit `options`, w którym znajduje się klauzula `directory`.

Dyrektywy `cache` i `primary` zostały zamienione na akapity `zone` z klauzulami `type`, wskazującymi odpowiednio na `hint` i `master`.

Pliki stref nie muszą być w żaden sposób modyfikowane. Ich składnia pozostaje bez zmian.

Nowa składnia pliku konfiguracyjnego pozwala na użycie wielu opcji, których tutaj nie omawialiśmy. Jeżeli szukasz informacji na ich temat, najlepszym źródłem jest dokumentacja dostarczana w pakiecie źródłowym BIND-a w wersji 8.

Pliki bazy danych DNS

Pliki główne związane z *named*, takie jak *named.hosts*, zawsze przynależą do jakiejś domeny, która nosi nazwę *początkowej* (ang. *origin*). Jest to nazwa domeny określona przez opcje `cache` i `primary`. W pliku głównym możesz podawać nazwy domen i hostów względem tej domeny. Nazwy podane w pliku konfiguracyjnym są traktowane jako *bezwzględne*, jeżeli kończą się kropką – w przeciwnym razie są one odnoszone do domeny *początkowej*. Do samej domeny początkowej można się odwoływać, używając znaku (@).

Dane zawarte w pliku głównym są podzielone na *rekordy zasobów* (ang. *resource records* – RR). RR są najmniejszymi jednostkami informacji dostępnymi przez DNS. Każdy rekord zasobu ma typ. Rekordy typu A na przykład wiążą nazwę hosta z adresem IP, a rekordy CNAME zawierają alias oficjalnej nazwy hosta. Możesz to zobaczyć w zamieszczonym dalej przykładzie 6-11, który pokazuje plik główny *named.hosts* dla browaru wirtualnego.

Reprezentacja rekordu zasobu w plikach głównych ma wspólny format:

```
[domena] [ttl] [klasa] typ danerek
```

Pola są oddzielane spacjami lub tabulatorami. Wpis może liczyć kilka wierszy, jeżeli przed pierwszym nowym wierszem i po ostatnim polu pojawią się nawiasy klamrowe. Wszystko pomiędzy średnikiem a znakiem nowego wiersza jest ignorowane. Oto opis pól:

`domena`
: Jest to nazwa domeny, której dotyczy wpis. Jeżeli nazwa domeny nie jest podana, uznaje się, że RR dotyczy domeny z poprzedniego RR.

`ttl`
: Aby wymusić na resolverach usuwanie informacji po pewnym czasie, z każdym RR jest związany czas życia (*ttl*). Pole ttl określa, w sekundach, czas ważności informacji, liczony od momentu jej uzyskania z serwera. Jest to liczba dziesiętna, maksymalnie ośmiocyfrowa.

  Jeżeli nie zostanie podana wartość ttl, przyjmowana jest domyślna wartość pola `minimum` poprzedniego rekordu SOA.

`klasa`
: Jest to klasa adresu, jak IN dla adresów IP czy HS dla obiektów z klasy Hesiod. W sieci TCP/IP musisz podawać IN.

  Jeżeli nie zostanie podane pole klasy, przyjmowana jest klasa z poprzedniego RR.

`typ`
: To pole opisuje typ RR. Najczęściej używane typy to A, SOA, PTR i NS. W dalszych podrozdziałach opisano różne typy RR.

danerek

To pole zawiera dane związane z RR. Format tego pola zależy od typu RR. W dalszej części opiszemy oddzielnie każdy RR.

Oto wybiórcza lista RR używanych w plikach głównych DNS. Istnieje jeszcze kilka innych rekordów, których nie będziemy tu opisywać, gdyż są albo eksperymentalne, albo rzadko używane:

SOA

Ten RR opisuje strefę władzy (SOA oznacza „początek władzy"). Sygnalizuje, że rekordy występujące po tym typie RR zawierają informacje autorytatywne dla domeny. Każdy plik główny wpisany w dyrektywie `primary` musi zawierać rekord SOA dla danej strefy. Dopuszczalne są następujące pola:

origin

To pole zawiera kanoniczną nazwę hosta podstawowego serwera nazw dla tej domeny, zwykle pisaną w postaci nazwy bezwzględnej.

contact

To pole to adres e-mail osoby odpowiedzialnej za utrzymanie domeny; w tym adresie znak @ został zastąpiony kropką. Na przykład, gdyby osobą odpowiedzialną za browar wirtualny była **janet**, pole miałoby postać `janet.vbrew.com`.

serial

To pole zawiera numer wersji pliku strefy wyrażony w postaci jednej liczby dziesiętnej. Gdy dane w pliku strefy zostaną zmienione, numer ten powinien zostać zwiększony. Przyjęło się, że numer ten to data ostatniego uaktualnienia z dodanym numerem wersji – na wypadek kilku uaktualnień w ciągu dnia. Na przykład 2000012600 oznacza uaktualnienie numer 00 z dnia 26 stycznia 2000 roku.

Numer jest używany przez zapasowe serwery nazw do rozpoznawania zmian w informacjach o strefie. Aby być na bieżąco, serwery zapasowe co jakiś czas odczytują rekord SOA serwerów podstawowych i porównują numer z własnym rekordem SOA. Jeżeli numer się zmienił, serwery zapasowe ściągają całą bazę danych z serwera podstawowego.

refresh

To pole określa w sekundach, co jaki czas serwery zapasowe powinny sprawdzać rekord SOA serwera podstawowego. Znów jest to liczba dziesiętna złożona maksymalnie z ośmiu cyfr.

Ogólnie rzecz biorąc topologia sieci nie zmienia się zbyt często, a więc ta liczba powinna być ustawiona na około jeden dzień w przypadku większych sieci, a nawet dłużej w przypadku mniejszych sieci.

retry

Ta liczba określa odstępy czasu, w których serwer zapasowy powinien próbować kontaktować się z serwerem podstawowym, jeżeli nie uda się mu odświeżyć danych o strefie. Nie może być ona zbyt mała, gdyż w razie chwilowej awarii ser-

ra lub sieci serwer zapasowy będzie marnował zasoby sieciowe. Zaleca się odstępy godzinne lub półgodzinne.

`expire`

To pole określa czas w sekundach, po którym serwer zapasowy powinien ostatecznie usunąć wszystkie dane o strefie, jeżeli nie był w stanie skontaktować się z serwerem głównym. Zwykle powinieneś zdefiniować to pole na przynajmniej tydzień (604 800 sekund), ale wydłużenie do miesiąca lub większe, także ma sens.

`minimum`

To pole zawiera domyślną wartość *ttl* dla rekordów zasobów, które nie definiują jej jawnie. Wartość *ttl* określa maksymalny czas, przez jaki inne serwery nazw mogą trzymać RR w swojej pamięci podręcznej. Czas ten dotyczy tylko zwykłych poszukiwań i nie ma nic wspólnego z czasem, po którym serwery zapasowe powinny próbować uaktualnić swoje informacje o strefie.

Jeżeli topologia twojej sieci nie zmienia się często, wystarczy, że ustawisz *ttl* na tydzień, a nawet dłuższy okres czasu. Jeżeli pojedynczy rekord RR zmienia się często, zawsze możesz przypisać mu indywidualnie mały *ttl*. Jeżeli twoja sieć zmienia się często, możesz zechcieć ustawić `minimum` na jeden dzień (86 400 sekund).

A

Ten rekord wiąże adres IP z nazwą hosta. Pole danych zasobu zawiera adres w notacji kropkowej.

Może istnieć tylko jeden rekord A dla danego hosta. Nazwa hosta używana w rekordzie A jest uznawana za oficjalną, inaczej *kanoniczną*, nazwę hosta. Wszystkie pozostałe nazwy hosta to aliasy, które muszą być odwzorowane na nazwę kanoniczną za pomocą rekordu CNAME. Jeżeli nazwa kanoniczna naszego hosta brzmiałaby **vlager**, mielibyśmy rekord A wiążący tę nazwę z adresem IP tego hosta. Ponieważ czasem chcemy związać z adresem także inną nazwę, powiedzmy **news**, mamy możliwość stworzenia rekordu CNAME, który wiąże nazwę alternatywną z nazwą kanoniczną. Więcej na temat rekordów CNAME powiemy już wkrótce.

NS

Rekordy NS są używane do określenia podstawowego serwera strefy i wszystkich jej serwerów zapasowych. Rekord NS wskazuje na główny serwer nazw danej strefy, a pole danych zasobu zawiera nazwę tego serwera.

Z rekordami NS spotkasz się w dwóch sytuacjach: po pierwsze, jeżeli przekazujesz władzę strefie podrzędnej, a po drugie, w głównej bazie danych samej strefy podrzędnej. Zestaw serwerów podanych w obu strefach (nadrzędnej i podrzędnej) powinien być taki sam.

Rekord NS określa nazwę podstawowego i zapasowego serwera nazw dla strefy. Nazwy te muszą być zamienione na adresy, aby można było z nich korzystać. Czasami serwery należą do domeny, którą obsługują, co rodzi problem „jajka i kury”. Nie możemy znaleźć adresu, dopóki serwer nazw jest nieosiągalny, ale

też nie możemy dotrzeć do serwera nazw, dopóki nie znajdziemy jego adresu. Aby rozwiązać ten dylemat, możemy skonfigurować specjalne rekordy A bezpośrednio w serwerze nazw strefy nadrzędnej. Rekordy A pozwalają serwerom nazw domeny nadrzędnej rozwiązywać adresy IP serwerów strefy podrzędnej. Te rekordy są powszechnie nazywane *rekordami klejącymi*, ponieważ dają możliwość powiązania strefy podrzędnej z jej strefą nadrzędną.

CNAME

Ten rekord wiąże alias z *kanoniczną* nazwą hosta. Udostępnia on alternatywną nazwę, za pomocą której użytkownicy mogą odwoływać się do hosta, którego nazwa kanoniczna jest podana jako parametr. Kanoniczna nazwa hosta to taka, dla której w pliku głównym istnieje rekord A. Aliasy są po prostu związane z tą nazwą poprzez rekord CNAME, ale nie mają innych własnych rekordów.

PTR

Ten typ rekordu jest używany do powiązania nazw w domenie **in-addr.arpa** z nazwami hostów. Służy do odwrotnego odwzorowania adresów IP na nazwy hostów. Podana nazwa hosta musi być nazwą kanoniczną.

MX

Ten RR określa *host wymieniający pocztę* (ang. *mail exchanger*) dla domeny. Hosty wymieniające pocztę są omówione w rozdziale 17, *Poczta elektroniczna*. Składnia rekordu MX jest następująca:

```
[domena] [ttl] [klasa] MX priorytet host
```

`Host` to nazwa hosta wymieniającego pocztę dla `domeny`. Z każdym hostem wymieniającym pocztę związany jest `priorytet`. Agent transportowy poczty, który chce dostarczyć pocztę do `domeny`, sprawdza wszystkie hosty, które dla danej domeny mają rekord MX, aż mu się uda z jakimś skontaktować. Najpierw jest sprawdzany host o najniższym priorytecie, a następnie po zostałe – w rosnącej kolejności priorytetów.

HINFO

Ten rekord zawiera informacje o sprzęcie i oprogramowaniu systemu. Składnia jest następująca:

```
[domena] [ttl] [klasa] HINFO sprz□t oprogramowanie
```

Pole `sprz□t` identyfikuje sprzęt używany w danym hoście. Do jego opisania stosowane są specjalne konwencje. Lista dopuszczalnych „nazw maszyn" jest podana w RFC *Assigned Numbers* (RFC-1700). Jeżeli pole zawiera spacje, musi być ujęte w cudzysłów. Pole `oprogramowanie` opisuje system operacyjny używany przez dany host. Znów dopuszczalne nazwy są opisane w RFC *Assigned Numbers*.

Rekord `HINFO` opisujący komputer oparty na procesorze Intel z zainstalowanym Linuksem powinien wyglądać następująco:

```
tao    36500    IN HINFO    IBM-PC    LINUX2.2
```

Natomiast rekord `HINFO` dla Linuksa działającego na komputerze z procesorem Motorola 68000 mógłby wyglądać tak:

```
cevad      36500    IN HINFO    ATARI-104ST  LINUX2.0
jedd       36500    IN HINFO    AMIGA-3000   LINUX2.0
```

Konfiguracja named jako serwera pamięci podręcznej

Istnieje szczególny typ konfiguracji *named*, który należy omówić, zanim wyjaśnimy, jak w pełni skonfigurować serwer nazw. Jest to konfiguracja *serwera pamięci podręcznej*. W rzeczywistości nie obsługuje on domeny, ale działa jako przekaźnik dla wszystkich zapytań DNS wygenerowanych przez hosty. Zaletą takiego schematu jest tworzenie pamięci podręcznej, a więc tylko pierwsze zapytanie o zadane hosty jest w rzeczywistości wysyłane do serwera nazw w Internecie. Odpowiedzi na wszelkie powtórne zapytania będą wysłane bezpośrednio z pamięci podręcznej twojego lokalnego serwera nazw. Może teraz nie wydaje się to zbyt użyteczne, ale zmienisz zdanie, jeżeli zaczniesz łączyć się z Internetem przez telefon, co opisano w rozdziale 7, *IP łącza szeregowego*, i rozdziale 8, *Protokół punkt-punkt*.

Plik *named.boot* dla serwera pamięci podręcznej wygląda następująco:

```
; plik named.boot dla serwera pami□ci podr□cznej
directory                              /var/named
primary      0.0.127.in-addr.arpa      named.local   ; sie□ hosta lokalnego
cache        .                         named.ca      ; serwery domeny g□ównej
```

Poza powyższym plikiem *named.boot*, musisz też skonfigurować plik *named.ca*, w którym będzie się znajdowała poprawna lista serwerów nazw domeny głównej. Możesz skopiować i wykorzystać do tego celu przykład 6-10. Do konfiguracji serwera nazw jako serwera pamięci podręcznej nie są potrzebne żadne inne pliki.

Tworzenie plików głównych

Przykłady od 6-10 do 6-13 pokazują przykładowe pliki serwera nazw sieci browaru, umieszczonego na hoście **vlager**. Ze względu na charakter omawianej sieci (pojedyncza sieć lokalna) przykład jest dość prosty.

Plik pamięci podręcznej *named.ca*, podany jako przykład 6-10, pokazuje przykładowe rekordy wskazujące serwer nazw domeny głównej. Typowy plik pamięci podręcznej zwykle zawiera listę kilku serwerów. Aktualną listę serwerów nazw domeny głównej możesz uzyskać za pomocą narzędzia *nslookup* opisanego w następnym podrozdziale*.

* Zauważ, że nie możesz zapytać serwera nazw o serwery nazw domeny głównej, jeżeli nie masz zainstalowanych żadnych wskazań na serwery domeny głównej. Aby rozwiązać ten problem, możesz ustawić *nslookup* tak, aby skorzystał z innego serwera nazw, albo użyć przykładowego pliku z przykładu 6-10 jako punktu wyjścia, a następnie uzyskać pełną listę dopuszczalnych serwerów.

Przykład 6-10. Plik na med.ca

```
;
; /var/named/named.ca   Plik pami□ci podr□cznej dla browaru.
;         Nie jeste□my pod□□czeni do Internetu, a wi□c
;         nie potrzebujemy □adnych serwerów nazw domeny
;         g□ównej. Aby uaktywni□ te rekordy, usu□ □redniki.
;
;.                     3600000   IN  NS     A.ROOT-SERVERS.NET.
;A.ROOT-SERVERS.NET.   3600000       A      198.41.0.4
;.                     3600000   IN  NS     B.ROOT-SERVERS.NET.
;B.ROOT-SERVERS.NET.   3600000       A      128.9.0.107
;.                     3600000   IN  NS     C.ROOT-SERVERS.NET.
;C.ROOT-SERVERS.NET.   3600000       A      192.33.4.12
;.                     3600000   IN  NS     D.ROOT-SERVERS.NET.
;D.ROOT-SERVERS.NET.   3600000       A      128.8.10.90
;.                     3600000   IN  NS     E.ROOT-SERVERS.NET.
;E.ROOT-SERVERS.NET.   3600000       A      192.203.230.10
;.                     3600000   IN  NS     F.ROOT-SERVERS.NET.
;F.ROOT-SERVERS.NET.   3600000       A      192.5.5.241
;.                     3600000   IN  NS     G.ROOT-SERVERS.NET.
;G.ROOT-SERVERS.NET.   3600000       A      192.112.36.4
;.                     3600000   IN  NS     H.ROOT-SERVERS.NET.
;H.ROOT-SERVERS.NET.   3600000       A      128.63.2.53
;.                     3600000   IN  NS     I.ROOT-SERVERS.NET.
;I.ROOT-SERVERS.NET.   3600000       A      192.36.148.17
;.                     3600000   IN  NS     J.ROOT-SERVERS.NET.
;J.ROOT-SERVERS.NET.   3600000       A      198.41.0.10
;.                     3600000   IN  NS     K.ROOT-SERVERS.NET.
;K.ROOT-SERVERS.NET.   3600000       A      193.0.14.129
;.                     3600000   IN  NS     L.ROOT-SERVERS.NET.
;L.ROOT-SERVERS.NET.   3600000       A      198.32.64.12
;.                     3600000   IN  NS     M.ROOT-SERVERS.NET.
;M.ROOT-SERVERS.NET.   3600000       A      202.12.27.33
;
```

Przykład 6-11. Plik na med.hosts

```
;
; /var/named/named.hosts      Hosty lokalne w browarze
;                              domena vbrew.com
;
@         IN SOA   vlager.vbrew.com. janet.vbrew.com {
                    2000012601 ; numer kolejny
                    86400      ; od□wie□anie:   raz dziennie
                    3600       ; ponowna próba: co godzin□
                    3600000    ; wyga□ni□cie:   42 godziny
                    604800     ; minimum:       1 tydzie□
                    }
           IN NS    vlager.vbrew.com.
;
; poczta lokalna jest dystrybuowana na vlager
           IN MX    10 vlager
;
; adres p□tli zwrotnej
localhost. IN A     127.0.0.1
;
; Ethernet browaru wirtualnego
vlager     IN A     172.16.1.1
vlager-if1   IN CNAME   vlager
; vlager to tak□e serwer grup dyskusyjnych
```

```
news         IN CNAME   vlager
vstout       IN A       172.16.1.2
vale         IN A       172.16.1.3
;
; Ethernet winiarni wirtualnej
vlager-if2   IN A       172.16.2.1
vbardolino   IN A       172.16.2.2
vchianti     IN A       172.16.2.3
vbeaujolais  IN A       172.16.2.4
;
; Ethernet wirtualnej fabryki napojów alkoholowych
; (dodatkowych)
vbourbon     IN A       172.16.3.1
vbourbon-if1 IN CNAME   vbourbon
```

Przykład 6-12. Plik named.local

```
;
; /var/named/named.local      Odwzorowanie odwrotne sieci 127.0.0
;                             domena pocz□tkowa 0.0.127.in-addr.arpa.
;
@            IN SOA   vlager.vbrew.com. joe.vbrew.com. {
                      1          ; numer kolejny
                      360000     ; od□wie□anie:   co 100 godzin
                      3600       ; ponowna próba: co godzin□
                      3600000    ; wyga□ni□cie:   42 dni
                      360000     ; minimum:       100 godzin
                      }
             IN NS    vlager.vbrew.com.
1            IN PTR   localhost.
```

Przykład 6-13. Plik named.rev

```
;
; /var/named/named.rev   Odwzorowanie odwrotne naszych adresów IP
;                        domena pocz□tkowa to 16.172.in-addr.arpa.
;
@            IN SOA   vlager.vbrew.com. joe.vbrew.com. {
                      16         ; numer kolejny
                      86400      ; od□wie□anie:   raz dziennie
                      3600       ; ponowna próba: co godzin□
                      3600000    ; wyga□ni□cie:   42 dni
                      604800     ; minimum:       1 tydzie□
                      }
             IN NS    vlager.vbrew.com.

; browar
1.1          IN PTR   vlager.vbrew.com.
2.1          IN PTR   vstout.vbrew.com.
3.1          IN PTR   vale.vbrew.com.
; winiarnia
1.2          IN PTR   vlager-if2.vbrew.com.
2.2          IN PTR   vbardolino.vbrew.com.
3.2          IN PTR   vchianti.vbrew.com.
4.2          IN PTR   vbeaujolais.vbrew.com.
```

Weryfikowanie konfiguracji serwera nazw

nslookup jest doskonałym narzędziem do sprawdzania działania twojego serwera nazw. Można go używać zarówno interaktywnie z monitem, jak i w formie polecenia natychmiast wypisującego wynik. W tym ostatnim przypadku po prostu wywołujesz polecenie tak:

```
$ nslookup nazwahosta
```

nslookup zadaje zapytanie o `nazwęhosta` do serwera nazw określonego w pliku *resolv.conf*. (Jeżeli w pliku znajduje się więcej niż jeden serwer, *nslookup* wybiera jakiś losowo).

Jednak tryb interaktywny jest dużo ciekawszy. Poza poszukiwaniem poszczególnych hostów, możesz zadawać zapytania o dowolny typ rekordu DNS i przesyłać całe informacje o strefie dla danej domeny.

Po wywołaniu *nslookup* bez argumentów, wyświetla on używany serwer nazw i przechodzi do trybu interaktywnego. Po monicie > możesz wpisać nazwę domeny, o którą chcesz pytać. Domyślnie zadawane są zapytania o klasę rekordów A – tych zawierających adres IP odnoszący się do nazwy domeny.

Innych typów rekordów możesz poszukać następująco:

```
> set type=typ
```

gdzie `typ` to jedna z nazw rekordu zasobu opisanych wcześniej lub dyrektywa ANY.

Można sobie wyobrazić następującą sesję z programem *nslookup*:

```
$ nslookup
Default Server: tao.linux.org.au
Address: 203.41.101.121

> metalab.unc.edu
Server: tao.linux.org.au
Address: 203.41.101.121
Name: metalab.unc.edu
Address: 152.2.254.81

>
```

Wynik najpierw pokazuje serwer DNS, do którego są kierowane zapytania, a następnie odpowiedź na zapytanie.

Jeżeli spróbujesz zapytać o nazwę, z którą nie jest związany adres IP, ale w bazie DNS znajdują się inne rekordy, *nslookup* zwróci komunikat o treści `No type A records found.` (nie znaleziono rekordów typu A). Jednak możesz zadać zapytanie nie tylko o rekordy A – trzeba tylko wydać polecenie *set type*. Aby uzyskać rekord SOA z domeny **unc.edu**, musisz napisać:

```
> unc.edu
Server: tao.linux.org.au
Address: 203.41.101.121

*** No address (A) records available for unc.edu
> set type=SOA
```

```
> unc.edu
Server: tao.linux.org.au
Address: 203.41.101.121

unc.edu
      origin = ns.unc.edu
      mail addr = host-reg.ns.unc.edu
      serial = 1998111011
      refresh = 14400 (4H)
      retry = 3600 (1H)
      expire = 1209600 (2W)
      minimum ttl = 86400 (1D)
unc.edu name server = ns2.unc.edu
unc.edu name server = ncnoc.ncren.net
unc.edu name server = ns.unc.edu
ns2.unc.edu       internet address = 152.2.253.100
ncnoc.ncren.net   internet address = 192.101.21.1
ncnoc.ncren.net   internet address = 128.109.193.1
ns.unc.edu        internet address = 152.2.21.1
```

W podobny sposób możesz zapytać o rekordy MX:

```
> set type=MX
> unc.edu
Server: tao.linux.org.au
Address: 203.41.101.121

unc.edu preference = 0, mail exchanger = conga.oit.unc.edu
unc.edu preference = 10, mail exchanger = imsety.oit.unc.edu
unc.edu name server = ns.unc.edu
unc.edu name server = ns2.unc.edu
unc.edu name server = ncnoc.ncren.net
conga.oit.unc.edu    internet address = 152.2.22.21
imsety.oit.unc.edu   internet address = 152.2.21.99
ns.unc.edu           internet address = 152.2.21.1
ns2.unc.edu          internet address = 152.2.253.100
ncnoc.ncren.net      internet address = 192.101.21.1
ncnoc.ncren.net      internet address = 128.109.193.1
```

Użycie typu ANY zwróci wszystkie rekordy za sobów związane z daną nazwą.

Innym praktycznym zastosowaniem *nslookup*, poza debugowaniem, jest uzyskiwanie aktualnej listy serwerów nazw domeny głównej. W tym celu należy zadać zapytanie o wszystkie rekordy NS związane z domeną główną.

```
> set type=NS
> .
Server: tao.linux.org.au
Address: 203.41.101.121

Non-authoritative answer:
(root)   name server = A.ROOT-SERVERS.NET
(root)   name server = H.ROOT-SERVERS.NET
(root)   name server = B.ROOT-SERVERS.NET
(root)   name server = C.ROOT-SERVERS.NET
(root)   name server = D.ROOT-SERVERS.NET
(root)   name server = E.ROOT-SERVERS.NET
(root)   name server = I.ROOT-SERVERS.NET
(root)   name server = F.ROOT-SERVERS.NET
(root)   name server = G.ROOT-SERVERS.NET
```

```
(root)   name server = J.ROOT-SERVERS.NET
(root)   name server = K.ROOT-SERVERS.NET
(root)   name server = L.ROOT-SERVERS.NET
(root)   name server = M.ROOT-SERVERS.NET

Authoritative answers can be found from:
A.ROOT-SERVERS.NET   internet address = 198.41.0.4
H.ROOT-SERVERS.NET   internet address = 128.63.2.53
B.ROOT-SERVERS.NET   internet address = 128.9.0.107
C.ROOT-SERVERS.NET   internet address = 192.33.4.12
D.ROOT-SERVERS.NET   internet address = 128.8.10.90
E.ROOT-SERVERS.NET   internet address = 192.203.230.10
I.ROOT-SERVERS.NET   internet address = 192.36.148.17
F.ROOT-SERVERS.NET   internet address = 192.5.5.241
G.ROOT-SERVERS.NET   internet address = 192.112.36.4
J.ROOT-SERVERS.NET   internet address = 198.41.0.10
K.ROOT-SERVERS.NET   internet address = 193.0.14.129
L.ROOT-SERVERS.NET   internet address = 198.32.64.12
M.ROOT-SERVERS.NET   internet address = 202.12.27.33
```

Aby zobaczyć pełny zestaw dostępnych poleceń, użyj w *nslookup* komendy *help*.

Inne przydatne narzędzia

Istnieje kilka narzędzi, które mogą pomóc w wypełnianiu obowiązków administratora BIND. Pokrótce opiszemy dwa z nich. Więcej informacji na ten temat znajdziesz w dołączonej do nich dokumentacji.

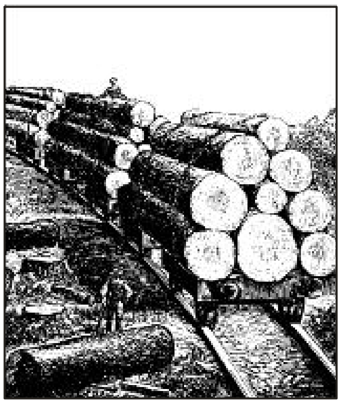
hostcvt pomaga we wstępnej konfiguracji BIND, konwertując plik */etc/hosts* na pliki główne dla *named*. Program generuje wpisy normalne (A) i odwrotne (PTR); obsługuje także aliasy. Oczywiście nie zrobi za ciebie wszystkiego, gdyż musisz chociażby dopasować wartości czasów w rekordzie SOA, czy dodać rekordy MX. Ale i tak zaoszczędzisz sobie kilku tabletek od bólu głowy. *hostcvt* jest częścią pakietu BIND, ale w linuksowych ośrodkach FTP można go znaleźć także w postaci samodzielnego programu.

Po skonfigurowaniu swojego serwera nazw, na pewno będziesz chciał sprawdzić jego konfigurację. Istnieją dobre narzędzia, które znacznie ułatwiają to zadanie: pierwszym z nich jest *dnswalk* – pakiet w języku Perl. Drugi nazywa się *nslint*. Oba programy przeglądają bazę DNS w poszukiwaniu popularnych błędów i weryfikują, czy znalezione informacje są spójne. Dwa inne przydatne narzędzia to: *host* i *dig* – są to narzędzia ogólnego przeznaczenia do zadawania zapytań do bazy DNS. Możesz je wykorzystać do ręcznego sprawdzania i diagnozowania wpisów w bazie danych DNS.

Wymienione narzędzia są dostępne w postaci pakietów. *dnswalk* i *nslint* są dostępne w postaci kodu źródłowego pod adresami: *http://www.visi.com/~barr/dnswalk/* i *ftp://ftp.ee.lbl.gov/nslint.tar.Z*. Kody źródłowe narzędzi *host* i *dig* można znaleźć pod adresami *ftp://ftp.nikhef.nl/pub/network/* i *ftp://ftp.is.co.za/networking/ip/dns/dig/*.

7

IP łącza szeregowego



Protokoły pakietowe, takie jak IP czy IPX, działają w oparciu o to, że host odbierający wie, gdzie się zaczyna i kończy każdy pakiet w strumieniu danych. Mechanizm używany do zaznaczania i wykrywania początku i końca pakietów nazywa się *rozgraniczaniem* (ang. *delimitation*). Za działanie tego mechanizmu w sieciach lokalnych odpowiada protokół Ethernet, natomiast w łączach szeregowych – protokoły SLIP i PPP.

Stosunkowo mały koszt wolnych modemów komutowanych i sieci telefonicznych spowodował, że protokoły IP łącza szeregowego stały się bardzo popularne, szczególnie do zapewniania łączności użytkownikom końcowym Internetu. Sprzęt potrzebny do uruchomienia SLIP czy PPP jest prosty i łatwo dostępny. Wystarczy mieć modem i port szeregowy z buforem FIFO.

Protokół SLIP jest bardzo prosty w implementacji i swego czasu był popularniejszy niż PPP. Obecnie jednak prawie każdy chętniej sięga po protokół PPP, który udostępnia bardziej wyrafinowane funkcje. Ważniejszym z nich przyjrzymy się później.

Linux obsługuje sterowniki dla protokołów SLIP i PPP oparte na jądrze. Sterowniki te, które powstały jakiś czas temu, są stabilne oraz niezawodne. W tym i następnym rozdziale omówimy oba protokoły i sposób ich konfiguracji.

Wymagania ogólne

Aby korzystać z protokołu SLIP lub PPP, musisz skonfigurować podstawowe funkcje sieciowe opisane w poprzednich rozdziałach, a także interfejs pętli zwrotnej i resolver. Przy podłączeniu do Internetu będziesz chciał korzystać z DNS-u. Możliwości wyboru są tu identyczne jak przy PPP: możesz zadawać zapytania DNS, albo przez łącze szeregowe, jeśli wcześniej skonfigurujesz w pliku */etc/resolv.conf* adres IP serwera nazw swojego dostawcy Internetu, albo konfigurując serwer nazw pamięci podręcznej zgodnie z opisem w rozdziale 6.

Działanie SLIP-a

Serwery IP podłączone do łącza komutowanego często udostępniają usługę SLIP przez specjalne konta użytkowników. Po zalogowaniu się na takie konto, nie dostajesz typowej powłoki, tylko uruchamiany jest program lub skrypt powłoki, który włącza sterownik SLIP serwera dla łącza szeregowego i konfiguruje odpowiedni interfejs sieciowy. Następnie musisz zrobić to samo po swojej stronie łącza.

W niektórych systemach operacyjnych sterownik SLIP jest programem działającym w przestrzeni użytkownika. W Linuksie jest to część jądra, co powoduje, że działa on dużo szybciej. Przedkość ta jednak wymaga, by łącze szeregowe było jawnie przełączone w tryb SLIP. To przełączenie jest realizowane przez specjalny protokół obsługi łącza tty, SLIPDISC. Gdy tty jest w trybie normalnego protokołu obsługi (DISC0), wymiania dane tylko z procesami użytkownika, używając zwykłych wywołań *read(2)* i *write(2)*, a sterownik SLIP nie jest w stanie ani zapisywać do tty, ani z niego odczytywać. W trybie SLIPDISC role się odwracają: teraz procesy przestrzeni użytkownika są blokowane przed zapisywaniem do tty lub odczytywaniem z niego, natomiast wszystkie dane przychodzące na port szeregowy są przekazywane bezpośrednio do sterownika SLIP.

Sam sterownik SLIP jest w stanie pracować z wieloma odmianami protokołu SLIP. Poza zwykłym SLIP-em rozumie także CSLIP, który realizuje tak zwaną kompresję nagłówków Van Jacobsona wychodzących pakietów IP (opisaną w RFC-1144). Kompresja ta znacznie poprawia przepustowość łącza podczas sesji interaktywnych. Istnieją także sześciobitowe wersje obu tych protokołów.

Prostym sposobem na przełączenie łącza szeregowego w tryb SLIP jest użycie narzędzia *slattach*. Załóżmy, że twój modem jest już podłączony pod */dev/ttyS3* i poprawnie zalogowałeś się do serwera SLIP. Teraz musisz wydać polecenie:

```
# slattach /dev/ttyS3 &
```

Narzędzie to przełączy protokół obsługi *ttyS3* na SLIPDISC i podłączy urządzenie do jednego z interfejsów sieciowych SLIP. Jeżeli jest to twoje pierwsze aktywne łącze SLIP, zostanie ono podłączone do *sl0*. Drugie byłoby podłączone do *sl1* i tak dalej. Aktualne jądra obsługują domyślnie maksymalnie 256 jednocześnie aktywnych łączy SLIP.

Domyślny protokół obsługi łącza wybierany przez *slattach* to CSLIP. Możesz wybrać dowolny inny, używając przełącznika *-p*. Aby użyć zwykłego SLIP-a (bez kompresji), wydaj polecenie:

```
# slattach -p slip /dev/ttyS3 &
```

Dostępne protokoły obsługi są wymienione w tabeli 7-1. Masz także do dyspozycji specjalny pseudoprotokół obsługi o nazwie `adaptive`, który powoduje, że jądro automatycznie wykrywa, jaka enkapsulacja SLIP jest włączona po drugiej stronie.

Tabela 7-1. Protokoły obsługi SLIP w Linuksie

Protokół obsługi	*Opis*
slip	Tradycyjna enkapsulacja SLIP.
cslip	Enkapsulacja SLIP z kompresją nagłówków Van Jacobsona.
slip6	Enkapsulacja SLIP z kodowaniem 6-bitowym. Metoda kodowania jest podobna do używanej przez polecenie *uuencode* i powoduje, że datagram SLIP jest konwertowany do drukowalnych znaków ASCII. Konwersja ta jest przydatna, jeżeli nie masz 8-bitowego łącza szeregowego.
cslip6	Enkapsulacja SLIP z kompresją nagłówków Van Jacobsona i kodowaniem 6-bitowym.
adaptive	Nie jest to typowy protokół obsługi, ale powoduje, że jądro próbuje zidentyfikować protokół używany na odległej maszynie i dopasować się do niego.

Zauważ, że musisz używać tej samej enkapsulacji co twój partner. Na przykład, jeżeli host **cowslip** używa CSLIP, ty także musisz go używać. Gdyby twoje połączenie SLIP nie działało, to przede wszystkim powinieneś sprawdzić, czy oba końce łącza uwzględniły używanie kompresji nagłówków. Jeżeli nie jesteś pewien, czego używa drugi koniec, spróbuj skonfigurować swój host na adaptive slip. Być może jądro prawidłowo odgadnie typ.

slattach pozwala ci na włączenie nie tylko SLIP-a, ale także innych protokołów wykorzystujących łącze szeregowe, takich jak PPP czy KISS (inny protokół używany przez fanów ham radio). Mimo to nie jest powszechnie stosowany, gdyż są lepsze narzędzia do obsługi tych protokołów. Szczegóły znajdziesz na stronie podręcznika elektronicznego *slattach(8)*.

Po przełączeniu łącza na sterownik SLIP, musisz skonfigurować interfejs sieciowy. Znów, robisz to za pomocą standardowych poleceń *ifconfig* i *route*. Załóżmy, że połączyliśmy się telefonicznie z hosta **vlager** do serwera o nazwie **cowslip**. Na hoście **vlager** powinieneś napisać:

```
# ifconfig sl0 vlager-slip pointopoint cowslip
# route add cowslip
# route add default gw cowslip
```

Pierwsze polecenie konfiguruje interfejs jako łącze punkt-punkt do **cowslip**, a następne dwa polecenia dodają trasę do **cowslip** i trasę domyślną wykorzystującą **cowslip** jako gateway.

Warto zwrócić uwagę na dwie rzeczy w wywołaniu *ifconfig*: opcję `pointopoint` określającą adres drugiego końca łącza punkt-punkt i wykorzystanie **vlager-slip** jako adresu lokalnego interfejsu SLIP.

Wspomnieliśmy, że dla łącza SLIP możesz użyć tego samego adresu, który przypisałeś interfejsowi Ethernet na hoście **vlager**. W tym przypadku **vlager-slip** mógłby być po prostu aliasem adresu **172.16.1.1**. Jednak możliwe jest również użycie zupełnie innego adresu dla łącza SLIP. Jedną z takich sytuacji jest sieć używająca niezarejestrowanego adresu IP sieci, tak jak się to dzieje w naszym wirtualnym browa-

rze. Powrócimy do tej sytuacji i opiszemy ją bardziej szczegółowo w następnym podrozdziale.

W pozostałej części tego rozdziału zawsze będziemy używać **vlager-slip**, odwołując się do adresu lokalnego interfejsu SLIP.

Przy wyłączaniu łącza SLIP powinieneś najpierw usunąć wszystkie trasy wiodące przez **cowslip** za pomocą polecenia *route* z opcją *del*, a następnie zamknąć interfejs i wysłać programowi *slattach* sygnał zawieszenia. Następnie musisz rozłączyć modem, używając ponownie programu swojego terminala:

```
# route del default
# route del cowslip
# ifconfig sl0 down
# kill -HUP 516
```

Pamiętaj, aby *516* zastąpić numerem ID procesu (pokazywanym w wyniku działania **ps ax**) polecenia *slattach* kontrolującego urządzenie slip, które chcesz zamknąć.

Korzystanie z sieci prywatnych

Pewnie pamiętasz z rozdziału 5, *Konfigurowanie sieci TCP/IP*, że browar wirtualny ma sieć IP wykorzystującą niezarejestrowane numery sieci zastrzeżone do użytku wewnętrznego. Pakiety kierowane z lub do tych sieci nie są rutowane do Internetu. Gdyby **vlager** łączył się z **cowslip** i działał jako ruter dla sieci browaru wirtualnego, hosty w sieci browaru nie mogłyby się bezpośrednio komunikować z prawdziwymi hostami z Internetu, ponieważ ich pakiety byłyby po cichu odrzucane przez pierwszy poważniejszy ruter.

Aby rozwiązać ten problem, skonfigurujemy **vlager** tak, aby działał jako swego rodzaju przeskaźnik udostępniający usługi internetowe. W świecie zewnętrznym będzie się on przedstawiał jako normalny host podłączony do Internetu za pośrednictwem protokołu SLIP, z zarejestrowanym adresem IP (prawdopodobnie przypisanym przez dostawcę usług, do którego należy **cowslip**). Każdy, kto zaloguje się do **vlagera**, może używać programów działających w trybie tekstowym, takich jak *ftp*, *telnet* czy nawet *lynx*, i za ich pomocą korzystać z Internetu. Każdy, kto należy do sieci lokalnej browaru wirtualnego, może zatem wywołać *telnet* i zalogować się do **vlagera**, a następnie używać na nim programów. Dla niektórych aplikacji mogą istnieć rozwiązania, które nie wymagają logowania się do **vlagera**. Na przykład w przypadku użytkowników WWW moglibyśmy na hoście **vlager** uruchomić tak zwany *serwer proxy*, który przekazywałby wszystkie żądania od użytkowników do odpowiednich serwerów.

Wymóg zalogowania się do **vlagera** celem korzystania z Internetu jest nieco niewygodny. Ale ma też swoje zalety. Po pierwsze, eliminuje konieczność rejestrowania się w sieci IP, co wymaga wypełniania papierów i jest kosztowne, a po drugie daje dodatkowe korzyści przy konfigurowaniu firewalla. Firewalle są dedykowanymi hostami, które dając ograniczony dostęp do Internetu użytkownikom twojej sieci lokalnej, równocześnie zabezpieczają twoje wewnętrzne hosty przed atakami ze świata zewnętrznego. Prosta konfiguracja firewalla została szczegółowo opisana w rozdziale 9, *Firewall TCP/IP*. W rozdziale 11, *Maskowanie IP i translacja adresów sieciowych*,

omawiamy funkcję Linuksa zwaną maskowaniem IP, która może być doskonałą alternatywą dla serwerów proxy.

Załóżmy, że browar ma przypisany adres IP **192.168.5.74** dla dostępu przez SLIP. Aby uzyskać omówioną powyżej konfigurację, musisz jedynie wprowadzić ten adres do pliku */etc/hosts* z nazwą **vlager-slip**. Procedura włączenia interfejsu SLIP pozostaje bez zmian.

Korzystanie z polecenia dip

Dotychczas opisane czynności nie były trudne. Niemniej jednak zapewne wolałbyś je zautomatyzować. Dużo lepiej byłoby mieć proste polecenie, które realizuje wszystkie wymienione kroki niezbędne do otworzenia urządzenia szeregowego, zadzwonienia do dostawcy, zalogowania się, włączenia protokołu obsługi SLIP i skonfigurowania interfejsu sieciowego. Takim poleceniem jest *dip*.

Nazwa *dip* to skrót od angielskiego terminu *dialup IP* (IP łącza komutowanego). Polecenie to zostało napisane przez Freda van Kempena i rozwinięte do sycznacznie przez wiele osób. Obecnie prawie wszyscy używają wersji `dip337p-uri`, która jest dołączana do większości współczesnych dystrybucji Linuksa, a także jest dostępna w archiwum FTP **metalab.unc.edu**.

dip udostępnia interpreter prostego języka skryptowego, który może obsłużyć modem, zmienić tryb SLIP i skonfigurować interfejsy. Język skryptowy jest wystarczająco silny, by sprostać większości konfiguracji.

Aby skonfigurować interfejs SLIP, *dip* potrzebuje przywilejów użytkownika root. Może cię kusić, aby nadać programowi *dip* prawo setuid **root**, tak by wszyscy użytkownicy (bez konieczności posiadania uprawnień roota) mogli dzwonić do serwera SLIP. Jest to bardzo niebezpieczne, ponieważ ustawienie fałszywych interfejsów i domyślnych tras za pomocą polecenia *dip* może uszkodzić routing w twojej sieci. Co gorsza, da to twoim użytkownikom możliwość podłączenia się do *dowolnego* serwera SLIP i wykonywania z twojej sieci niebezpiecznych ataków. Jeśli chcesz pozwolić użytkownikom na uruchamianie połączenia SLIP, napisz małe programy dodatkowe dla każdego potencjalnego serwera SLIP i z nich wywołuj *dip* ze specjalnym skryptem nawiązującym połączenie. Poprawnie napisanemu programowi tego typu można bezpiecznie nadać prawo setuid **root***. Alternatywnym, bardziej elastycznym podejściem jest nadanie zaufanym użytkownikom uprawnień roota do *dip* poprzez program typu *sudo*.

Przykładowy skrypt

Załóżmy, że host, z którym łączysz się przez SLIP, to **cowslip**. Napisaliśmy skrypt dla *dipa* – *cowslip.dip* – który realizuje nasze połączenie. Wywołujemy *dip* z nazwą skryptu jako argumentem:

* *diplogin* też musi być uruchamiany z prawem setuid **root**, jak dowiesz się jeszcze z tego rozdziału.

```
# dip cowslip.dip
DIP: Dialup IP Protocol Driver version 3.3.7 (12/13/93)
Written by Fred N. van Kempen, MicroWalt Corporation.
connected to cowslip.moo.com with addr 192.168.5.74
#
```

Sam skrypt po kaz ano w przykładzie 7-1.

Przykład 7-1: Przykład o wy skrypt dip

```
# Przyk□adowy skrypt dip dzwoni□cy do cowslip
# Ustawienie lokalnych i zdalnych nazw i adresów
   get $local vlager-slip
   get $remote cowslip
   port ttyS3        # wybór portu szeregowego
   speed 38400      # ustawienie pr□dko□ci na maksimum
   modem HAYES      # ustawienie typu modemu
   reset            # wyzerowanie modemu i tty
   flush            # wyczyszczenie bufora modemu
# Przygotowanie do dzwonienia
   send ATQ0V1E1X1\r
   wait OK 2
   if $errlvl !=0 goto error
   dial 41988
   if $errlvl !=0 goto error
   wait CONNECT 60
   if $errlvl !=0 goto error
# Ok, jeste□my pod□□czeni
   sleep 3
   send \r\n\r\n
   wait ogin: 10
   if $errlvl !=0 goto error
   send $vlager\n
   wait ssword: 5
   if $errlvl !=0 goto error
   send knockknock\n
   wait running 30
   if $errlvl !=0 goto error
# Zalogowali□my si□ i po drugiej stronie uruchamiany jest SLIP.
   print Connected to $remote with address $rmtip
   default          # Ustawienie rutingu domy□lnego na to □□cze
   mode SLIP        # Przechodzimy tak□e do trybu SLIP
# tu trafiamy w razie wyst□pienia b□□du
error:
   print SLIP to $remote failed.
```

Po podłącze niu do **cowslip** i włącze niu SLIP, *dip* odłącza się od ter mi na la i prze cho- dzi do pra cy w tle. Możesz zacząć uru cha miać nor mal ne usługi sie cio we na łączu SLIP. Aby zakończyć połączenie, po prostu wy wołaj *dip* z opcją *-k*, co spowoduje wysłanie sy gnału do *dip*. Wy ko rzy sta ne zo sta nie do te go ID pro ce su, które *dip* za pi su- je w pli ku */etc/dip.pid*.

```
# dip -k
```

W ję zy ku skryp to wym po le ce nia *dip* słowa klu czo we po prze dzo ne zna kiem do la ra ozna czają na zwy zmien nych. *dip* posia da pre de fi nio wa ny ze staw zmien nych, które zo staną omó wio ne po niż ej. Na przykład `$remote` i `$local` za wie rają od po wied-

nio nazwy hostów lokalnego i zdalnego, znajdujących się po obu stronach łącza SLIP.

Pierwsze dwie dyrektywy w przykładowym skrypcie to polecenia *get*, za pomocą których *dip* definiuje zmienne. Nazwy hostów lokalnego i zdalnego są tu ustawiane odpowiednio na **vlager** i **cowslip**.

Następne pięć dyrektyw konfiguruje terminal i modem. `reset` wysyła do modemu ciąg zerujący. Kolejna dyrektywa czyści bufor modemu tak, aby dialog logowania umieszczony w kolejnych kilku wierszach zadziałał poprawnie. Dialog ten jest dosyć prosty: po prostu dzwoni pod numer 41 988 (numer telefonu **cowslip**) i loguje się na konto `$vlager` za pomocą hasła `knockknock`. Polecenie `wait` powoduje, że *dip* czeka na ciąg znaków podany jako pierwszy argument. Liczba podana jako drugi argument określa, po ilu sekundach kończy się oczekiwanie, jeżeli dany ciąg nie zostanie odebrany. Polecenia `if` występujące w procedurze logowania sprawdzają, czy nie wystąpił błąd w wykonywanym poleceniu.

Ostatnie polecenia wywoływane po zalogowaniu się to: `default`, ustawiające domyślny ruting na zestawione właśnie łącze SLIP, i `mode`, włączające tryb SLIP na łączu i konfigurujące interfejs oraz tablicę rutingu.

dip

W tym podrozdziale podamy opis większości poleceń *dip*. Listę wszystkich poleceń możesz zobaczyć, wywołując *dip* w trybie testowym i wprowadzając `help`. Aby poznać składnię polecenia, możesz je wprowadzić bez żadnych argumentów. Pamiętaj, że nie sprawdzisz tak składni tych poleceń, które nie potrzebują argumentów. Poniższy przykład ilustruje polecenie `help`:

```
# dip -t
DIP: Dialup IP Protocol Driver version 3.3.7p-uri (25 Dec 96)
Written by Fred N. van Kempen, MicroWalt Corporation.
Debian version 3.3.7p-2 (debian).

DIP> help
DIP knows about the following commands:

   beep       bootp      break      chatkey   config
   databits   dec        default    dial      echo
   flush      get        goto       help      if
   inc        init       mode       modem     netmask
   onexit     parity     password   proxyarp  print
   psend      port       quit       reset     secureidfixed
   securid    send       shell      skey      sleep
   speed      stopbits   term       timeout   wait

DIP> echo
Usage: echo on|off
DIP>
```

W kolejnych podrozdziałach przykłady zawierające monit *DIP>* pokazują, jak wprowadzać polecenia w trybie testowym i jaki dają one wynik. Przykłady bez tego monitu należy traktować jako fragmenty skryptów.

Polecenia dotyczące modemu

dip udostępnia szereg poleceń konfigurujących łącze szeregowe i modem. Niektóre z nich są oczywiste, np. `port`, służący do wyboru portu szeregowego, czy `speed`, `databits`, `stopbits` i `parity`, ustawiające typowe parametry łącza. Polecenie `modem` wybiera typ modemu. Aktualnie jedynym obsługiwanym trybem jest `HAYES` (wymagane pisanie dużymi literami). Musisz określić programowi *dip* typ modemu, gdyż w przeciwnym razie nie będzie możliwe wykonanie poleceń `dial` i `reset`. Polecenie `reset` wysyła ciąg znaków zerujący modem. Stosowany ciąg zależy od wybranego typu modemu. W przypadku modemów kompatybilnych ze standardem Hayesa, ciągiem tym jest `ATZ`.

Polecenie `flush` może być wykorzystane do usunięcia z bufora wszystkich odpowiedzi wysłanych przez modem do tej pory. W przeciwnym razie skrypt dialogowy występujący po poleceniu `reset` mógłby zgłupieć, ponieważ odczytałby odpowiedzi `OK` z poprzednich poleceń.

Polecenie `init` określa ciąg inicjacyjny przekazywany do modemu przed rozpoczęciem dzwonienia. Domyślny ciąg dla modemów Hayesa to `ATE0 Q0 V1 X1`, włącza on wypisywanie poleceń i dzwonienie na ślepo (bez sprawdzania sygnału na linii). Nowsze modemy posiadają dobrą konfigurację fabryczną, a więc jest to zbędne, choć w niczym nie przeszkadza.

Polecenie `dial` wysyła ciąg inicjacyjny do modemu i dzwoni do zdalnego systemu. Domyślne polecenie dzwonienia dla modemów Hayesa to `ATD`.

Polecenie echo

Polecenie `echo` jest doskonałą pomocą przy debugowaniu. Wywołanie `echo on` powoduje, że *dip* powtarza na konsoli wszystko, co wysyła do urządzenia szeregowego. Można to z powrotem wyłączyć, wydając polecenie `echo off`.

dip pozwala także na chwilę wyjść z trybu skryptowego i wejść do trybu terminalowego. W tym trybie używasz *dip* tak jak zwykłego programu terminala, wysyłając wpisywane znaki na łącze szeregowe, odczytując dane z łącza szeregowego i wyświetlając znaki. Aby wyjść z tego trybu, naciśnij [CTRL+]].

Polecenie get

Za pomocą polecenia `get` *dip* nadaje wartość zmiennej. Najprostsze jest przypisanie zmiennej stałej wartości, co robiliśmy w skrypcie *cowslip.dip*. Możesz jednak poprosić użytkownika o wprowadzenie czegoś, używając słowa kluczowego `ask` zamiast wartości:

```
DIP> get $local ask
Enter the value for $local: _
```

Trzecia metoda to uzyskanie wartości ze zdalnego hosta. W pierwszej chwili wygląda to dziwacznie, ale nieraz się bardzo przydaje. Niektóre serwery SLIP nie pozwolą ci używać własnego adresu IP na łączu SLIP, a po połączeniu będą przypisywać ci adres z puli, wypisując komunikat informujący o otrzymanym adresie. Jeżeli

otrzymasz komunikat: „`Your address: 192.168.5.74`, poniższy fragment kodu *dip* pozwoli ci przypisać ten adres:

```
# zako□czenie logowania si□
wait address: 10
get $locip remote
```

Polecenie print

Jest to polecenie używane do wyświetlania tekstów *dip* na konsoli, z której został uruchomiony. W poleceniu *print* mogą być użyte wszystkie zmienne *dip*. Oto przykład:

```
DIP> print Using port $port at speed $speed
Using port ttyS3 at speed 38400
```

Nazwy zmiennych

dip rozumie tylko predefiniowany zestaw zmiennych. Nazwy zmiennych zawsze zaczynają się od znaku dolara i muszą być pisane małymi literami.

Zmienne `$local` i `$locip` zawierają nazwę i adres lokalnego hosta. Jeżeli w zmiennej `$local` zapiszesz kanoniczną nazwę hosta, *dip* będzie automatycznie próbował zamienić nazwę hosta na jego adres IP i zapisać go w zmiennej `$locip`. Podobny, aczkolwiek odwrotny proces zachodzi, gdy przypiszesz adres IP zmiennej `$locip`: *dip* będzie próbował wyszukiwania odwrotnego, czyli będzie chciał zidentyfikować nazwę hosta i zapisać ją w zmiennej `$local`.

Zmienne `$remote` i `$rmtip` działają w ten sam sposób dla adresu i nazwy hosta zdalnego. `$mtu` zawiera wartość MTU dla połączenia.

Te pięć zmiennych to jedyne zmienne, którym wartości mogą być przypisywane bezpośrednio za pomocą polecenia *get*. Szereg innych zmiennych jest ustawianych w wyniku działania poleceń konfiguracyjnych o tej samej nazwie, a mogą one być używane w dyrektywie *print*. Należą do nich `$modem`, `$port` i `$speed`.

`$errlvl` jest zmienną, przez którą uzyskujesz wynik wykonania ostatniego polecenia. Poziom błędu 0 oznacza poprawne wykonanie, natomiast wartości różne od zera oznaczają błąd.

Polecenia if i goto

Polecenie `if` to tradycyjne rozgałęzienie warunkowe, a nie w pełni wyposażona programistyczna dyrektywa *if*. Składnia jest następująca:

```
if zm op liczba goto etykieta
```

Wyrażenie musi być prostym porównaniem jednej ze zmiennych: `$errlvl`, `$locip` lub `$rmtip`. *zm* musi być liczbą całkowitą, operator *op* może być jednym ze znaków ==, !=, <, >, <= i >=.

Polecenie `goto` powoduje, że wykonywanie skryptu jest kontynuowane od wiersza, który następuje po *etykiecie*. Etykieta musi być pierwszym słowem w wierszu i musi być zakończona dwukropkiem.

send, wait i sleep

Te polecenia pomagają zaimplementować proste skrypty dialogowe w *dip*. Polecenie `send` wysyła argumenty do łącza szeregowego. Nie obsługuje zmiennych, ale rozumie wszelkie sekwencje ze znakami ukośnika odwrotnego znane z języka C, takie jak `\n` oznaczające nowy wiersz i `\b` oznaczające cofnięcie. Znak tyldy (~) może być zastosowany jako skrót ciągu znaków: powrót karetki / nowy wiersz.

Polecenie `wait` jako argument przyjmuje słowo i odczytuje wszystkie dane wejściowe na łączu szeregowym, aż wykryje ciąg znaków pasujący do zadanego słowa. Samo słowo nie może zawierać spacji. Opcjonalnie, jako drugi argument, możesz podać w poleceniu `wait` wartość czasu oczekiwania. Jeżeli oczekiwane słowo nie zostanie odebrane w tym czasie, polecenie zwróci w zmiennej `$errlvl` wartość 1. Polecenie to jest używane do wykrywania monitu logowania i innych.

Polecenie `sleep` może być używane do odczekania pewnego czasu. Na przykład, aby cierpliwie zaczekać na zakończenie sekwencji logowania. Znów czas jest podawany w sekundach.

mode i default

Te polecenia są używane do przełączania łącza szeregowego w tryb SLIP i konfigurowania interfejsu.

Polecenie `mode` jest ostatnim poleceniem wykonywanym przez *dip* przed przejściem w tryb demona. Dopóki nie wystąpi błąd, polecenie nie kończy się.

`mode` przyjmuje jako argument nazwę protokołu. Obecnie *dip* rozpoznaje `SLIP`, `CSLIP`, `SLIP6`, `CSLIP6`, `PPP` i `TERM`. Jednak aktualna wersja *dip* nie rozumie trybu adaptive SLIP.

Po przełączeniu łącza szeregowego do trybu SLIP, *dip* wywołuje polecenie *ifconfig* w celu skonfigurowania interfejsu jako łącza punkt-punkt i polecenie *route* do skonfigurowania rutingu do zdalnego hosta.

Jeżeli skrypt dodatkowo wywoła polecenie `default` przed poleceniem `mode`, *dip* tworzy domyślny ruting wskazujący na łącze SLIP.

Działanie w trybie serwera

Skonfigurowanie klienta SLIP było trudniejszym zadaniem. Skonfigurowanie twojego hosta, aby działał jako serwer SLIP jest dużo łatwiejsze.

Istnieją dwa sposoby skonfigurowania serwera SLIP. Oba wymagają utworzenia jednego konta logowania dla każdego klienta SLIP. Załóżmy, że udostępniasz usługę SLIP Arthurowi Dentowi z **dent.beta.com**. Dodając poniższy wiersz do twojego pliku *passwd*, możesz stworzyć konto o nazwie **dent**:

```
dent:*:501:60:Konto SLIP Arthura Denta:/tmp:/usr/sbin/diplogin
```

Następnie za pomocą polecenia *passwd* musisz ustawić hasło użytkownika **dent**.

Polecenie *dip* może być uruchomione w trybie serwera przez wywołanie *diplogin*. Zwykle *diplogin* jest dowiązaniem do *dip*. Jego głównym plikiem konfiguracyjnym

jest */etc/diphosts*, w którym zwykle wpisujesz adres IP przypisywany użytkownikowi, gdy zadzwoni. Alternatywnie możesz także użyć polecenia *sliplogin*, narzędzia pochodzącego z BSD i posiadającego bardziej elastyczny schemat konfiguracji pozwalający ci na wywoływanie skryptów powłoki, gdy host się podłącza lub rozłącza.

Gdy nasz użytkownik SLIP-a, **dent**, zaloguje się, *dip* jest uruchamiany jako serwer. Aby stwierdzić, czy dany użytkownik naprawdę ma prawo używać SLIP-a, szuka on nazwy użytkownika w pliku */etc/diphosts*. Plik ten zawiera szczegółowe prawa dostępu i parametry połączenia dla każdego użytkownika SLIP-a. Ogólny format wpisu w */etc/diphosts* jest następujący:

```
# /etc/diphosts
u□ytkownik:has□o:adres-zdalny:adres-lokalny:maska:komentarze:protokó□,MTU
#
```

Każde z pól jest opisane w tabeli 7-2.

Tabela 7-2. Opis pól pliku /etc/diphosts

Pole	*Opis*
u□ytkownik	Pole to określa nazwę użytkownika wywołującego *dip*.
Has□o	Drugie pole pliku */etc/diphosts* jest używane do zapewnienia dodatkowej warstwy bezpieczeństwa przy łączeniu się na podstawie hasła. Możesz tu umieścić hasło w zaszyfrowanej postaci (tak jak w pliku */etc/passwd*), a *diplogin* poprosi użytkownika o wprowadzenie hasła, zanim pozwoli mu na dostęp SLIP. Zauważ, że jest to hasło uzupełniające, używane obok zwykłego hasła wprowadzanego w monicie *login*.
adres-zdalny	Adres, który zostanie przypisany zdalnej maszynie. Adres ten może być podany także w postaci nazwy hosta, która zostanie zamieniona na numer IP, albo bezpośrednio w postaci numeru IP w notacji kropkowej.
adres-lokalny	Adres IP, który będzie używany dla lokalnego końca połączenia SLIP. Może być podany w postaci nazwy hosta lub numeru IP.
maska	Maska sieci, która będzie używana do rutingu. Wiele osób źle interpretuje to pole. Maska sieci nie dotyczy samego łącza SLIP, ale jest używana w połączeniu z `adresem-zdalnym` do utworzenia trasy do zdalnej sieci. Maska sieci powinna być taka sama jak maska używana przez sieć obsługiwaną przez zdalnego hosta.
komentarze	Jest to pole tekstowe, w którym można wprowadzić dowolny opis i jest ono traktowane jako pomoc w dokumentowaniu pliku */etc/diphosts*. Pole to nie ma innych zastosowań.
protokó□	W tym polu określa się, jakiego protokołu obsługi chcesz używać dla danego połączenia. Poprawne wpisy są identyczne z używanymi z argumentem *-p* polecenia *slattach*.
MTU	Maksymalna jednostka transmisji, którą można przesłać przez łącze. To pole opisuje największy datagram przesyłany przez łącze. Każdy datagram rutowany przez urządzenie SLIP, który jest większy niż MTU, zostanie podzielony na datagramy nie większe niż ta wartość. MTU zwykle jest konfigurowane tak samo na obu końcach łącza.

Przykładowy wpis dla **dent** mógłby wyglądać tak:

```
dent::dent.beta.com:vbrew.com:255.255.255.0:Arthur Dent:CSLIP,296
```

Ten przykład daje naszemu użytkownikowi **dent** dostęp do SLIP-a bez potrzeby wprowadzania dodatkowego hasła. Będzie mu przypisany adres IP związany z nazwą **dent.beta.com** i maska sieci **255.255.255.0**. Domyślny ruting powinien być przekierowany na adres IP **vbrew.com**. Połączenie będzie wykorzystywało protokół CSLIP z MTU równym 296 bajtów.

Gdy **dent** się zaloguje, *diplogin* odczyta informacje na jego temat z pliku *diphosts*. Gdyby drugie pole zawierało wartość, *diplogin* poprosiłby o dodatkowe hasło. Wprowadzony przez użytkownika ciąg znaków zostałby zaszyfrowany i porównany z hasłem z pliku *diphosts*. Gdyby ciągi się nie zgadzały, próba zalogowania nie powiodłaby się. Gdyby pole hasła zawierało ciąg znaków `s/key`, a *dip* byłby skompilowany z obsługą S/Key, uruchomione byłoby uwierzytelnianie S/Key. Jest ono opisane w dokumentacji zawartej w pakiecie źródłowym *dip*.

Po poprawnym zalogowaniu się, *diplogin* przełącza łącze szeregowe w tryb CSLIP lub SLIP i konfiguruje interfejs oraz ruting. Połączenie pozostaje zestawione, dopóki użytkownik go nie rozłączy i modem nie zostanie odłączony od linii telefonicznej. *diplogin* przywraca łącze szeregowe do normalnego protokołu obsługi i kończy pracę.

diplogin wymaga praw użytkownika uprzywilejowanego. Jeżeli *dip* nie działa z prawem setuid **root**, powinieneś spowodować, żeby *diplogin* był oddzielną kopią *dip*, a nie dowiązaniem. *diplogin* może wtedy mieć nadane prawo setuid bez zmiany statusu samego *dip*.



8

Protokół punkt-punkt

Podobnie jak SLIP, protokół PPP jest używany do wysyłania datagramów przez łącze szeregowe, jednak nie ma on wielu wad SLIP-a. Po pierwsze, pozwala na przesyłanie większej liczby protokołów i nie jest ograniczony do protokołu IP. Ma możliwość wykrywania błędów na samym łączu, gdzie SLIP akceptował i przekazywał uszkodzone datagramy, chyba że uszkodzony został nagłówek. Ponadto, pozwala stronom połączenia negocjować na początku opcje, takie jak adres IP i maksymalny rozmiar datagramu, oraz zapewnia uwierzytelnianie klienta. Taka wbudowana możliwość negocjacji pozwala na niezawodną automatyzację przy zestawianiu połączenia, natomiast dzięki uwierzytelnianiu nie są potrzebne sztuczne kontа użytkowników, które były stosowane w przypadku SLIP-a. Każda z tych możliwości jest w PPP obsługiwana przez oddzielny protokół. W tym rozdziale krótko omówimy podstawowe moduły PPP. Niniejszy opis PPP jest daleki od kompletności, a więc jeżeli chcesz wiedzieć więcej, zachęcamy cię do przeczytania specyfikacji protokołu w odpowiednim dokumencie RFC i szeregu uzupełniających RFC. Istnieje również cała książka poświęcona temu tematowi: *Using & Managing PPP* napisana przez Andrew Suna (O'Reilly).

Na samym dole PPP znajduje się protokół wysokopoziomowego sterowania łączem danych (*High-Level Data Link Control* – HDLC), który definiuje granice pojedynczych ramek PPP i zapewnia 16-bitową sumę kontrolną*. W przeciwieństwie dość prymitywnej enkapsulacji SLIP, ramka PPP może zawierać pakiety różnych protokołów, nie tylko IP, czyli na przykład IPX Novella czy AppleTalk. PPP dodaje bowiem do podstawowej ramki HDLC pole protokołu identyfikujące typ pakietu przesyłanego w ramce.

Nad HDLC znajduje się *protokół sterowania łączem* (*Link Control Protocol* – LCP) negocjujący opcje dotyczące łącza danych, na przykład *maksymalną jednostkę odbioru* (*Maximum Receive Unit* – MRU) wyznaczającą maksymalny rozmiar datagramu, jaki jedna strona łącza zgodziła się odbierać.

---

* W rzeczywistości, HDLC jest protokołem ogólnego przeznaczenia stworzonym przez międzynarodową organizację standaryzacyjną ISO; jest on również istotnym składnikiem specyfikacji X.25.

Ważnym krokiem naprzód w konfiguracji łącza PPP jest autoryzacja (uwierzytelnienia) klienta. Choć nie jest ona obowiązkowa, powinna być używana w przypadku linii komutowanych, aby nie dopuścić intruzów do systemu. Zazwyczaj wywoływany host (serwer) prosi klienta o podanie tajnego klucza. Jeżeli host wywołujący nie wygeneruje odpowiedniego klucza, połączenie jest zrywane. W PPP autoryzacja działa w obie strony. Host wywołujący może również poprosić o autoryzację serwera. Te procedury są zupełnie niezależne od siebie. Dla dwóch różnych sposobów autoryzacji istnieją dwa protokoły, które będziemy dokładniej omawiali w tym rozdziale: *protokół uwierzytelniania hasłem* (*Password Authentication Protocol* – PAP) i *protokół uwierzytelnienia przez uzgodnienie* (*Challenge Handshake Authentication Protocol* — CHAP).

Każdy protokół sieciowy rutowany przez łącze danych (jak IP i AppleTalk) jest konfigurowany dynamicznie za pomocą odpowiedniego *protokołu sterowania siecią* (*Network Control Protocol* – NCP). Aby wysłać datagram IP przez łącze, obie strony uczestniczące w połączeniu PPP muszą najpierw wynegocjować używane przez każdą z nich adresy IP. Protokół sterujący używany w tej negocjacji to *protokół sterowania protokołem internetowym* (*Internet Protocol Control Protocol* – IPCP).

Poza wysyłaniem standardowych datagramów IP przez łącze, PPP także obsługuje kompresję nagłówków (Van Jacobsona) datagramów IP. Technika ta zmniejsza nagłówki pakietów IP do zaledwie trzech bajtów. Jest ona także stosowana w CSLIP i potocznie nazywa się ją kompresją nagłówków VJ. Użycie kompresji może być również negocjowane za pomocą protokołu IPCP.

PPP w Linuksie

W Linuksie do funkcjonowania PPP są potrzebne dwie rzeczy: element jądra obsługujący protokoły niskopoziomowe (HDLC, IPCP, IPXCP itp.) i demon *pppd* działający w przestrzeni użytkownika i obsługujący różne protokoły wyższego poziomu, takie jak PAP i CHAP. Aktualna wersja oprogramowania PPP dla Linuksa zawiera demon *pppd* i program o nazwie *chat*, które automatyzują połączenie telefoniczne z systemu zdalnego.

Sterownik PPP jądra został napisany przez Michaela Callahana i przerobiony przez Paula Mackerrasa. *pppd* powstało na podstawie darmowej implementacji PPP* dla Suna i komputerów 386BSD, napisanej przez Drew Perkinsa i innych i utrzymywanej przez Paula Mackerrasa. Zostało przeniesione na Linuksa przez Ala Longyeara. Program *chat* napisał Karl Fox**.

Podobnie jak SLIP, tak i PPP został zaimplementowany przez specjalny protokół obsługi łącza. Aby wykorzystać łącze szeregowe jako łącze PPP, musisz najpierw jak zwykle zestawić połączenie przez modem, a następnie przełączyć łącze w tryb PPP. W tym trybie wszystkie przychodzące dane są przekazywane sterownikowi PPP,

* Jeżeli masz jakiekolwiek ogólne pytania na temat PPP, kieruj je do listy dyskusyjnej *Linux-net* na adres **vger.rutgers.edu**.

** Z Karlem można się skontaktować pod adresem *karl@morningstar.com*.

który sprawdza poprawność przychodzących ramek HDLC (każda ramka HDLC zawiera 16-bitową sumę kontrolną), rozpakowuje je oraz obsługuje. Obecnie PPP jest w stanie przesyłać zarówno protokół IP, opcjonalnie z kompresją nagłówków Van Jacobsona, jak i protokół IPX.

pppd wspomaga sterownik jądra, wykonując obowiązkową fazę inicjacyjną i uwierzytelniającą, zanim rzeczywisty ruch sieciowy zostanie przesłany przez łącze. Zachowanie *pppd* można regulować szeregiem opcji. Ponieważ PPP jest raczej złożonym protokołem, nie możliwe jest wyjaśnienie wszystkich opcji w jednym rozdziale. Dlatego ta książka nie omawia wyczerpująco *pppd*, a tylko podaje ogólny zarys. Dokładniejsze informacje znajdziesz we wspomnianej już książce *Using & Managing PPP* lub na stronach podręcznika elektronicznego *pppd* oraz we wspomnianych już plikach *README* pakietu źródłowego *pppd*. Pomogą ci one odpowiedzieć na większość pytań, które tu taj nie zostały uwzględnione. Pomocny może być także dokument *PPP-HOWTO*.

Prawdopodobnie w konfigurowaniu PPP najbardziej pomogą ci inni użytkownicy tej samej dystrybucji Linuksa. Pytania o konfigurację PPP są dosyć powszechne, a więc sprawdź swoją lokalną grupę dyskusyjną lub kanał linuksowy na IRC-u. Jeżeli masz problemy nawet po przeczytaniu dokumentacji, możesz spróbować je rozwiązać poprzez grupę dyskusyjną *comp.protocols.ppp*. Jest to miejsce, gdzie spotyka się większość osób zaangażowanych w rozwój *pppd*.

Eksploatacja pppd

Gdy chcesz się podłączyć do Internetu przez łącze PPP, musisz skonfigurować podstawowe funkcje sieciowe, jak urządzenie pętli zwrotnej i resolver. Obie zostały omówione w rozdziale 5, *Konfigurowanie sieci TCP/IP*, i w rozdziale 6, *Usługi nazewnicze i konfigurowanie resolvera*. Możesz po prostu skonfigurować serwer nazw swojego dostawcy Internetu w pliku */etc/resolv.conf*, ale będzie to oznaczało, że każde żądanie DNS jest wysyłane przez łącze szeregowe. Ta sytuacja nie jest optymalna. Im jesteś bliżej (w sensie sieci) swojego serwera nazw, tym szybciej są realizowane wyszukiwania nazw. Alternatywnym rozwiązaniem jest skonfigurowanie serwera nazw pamięci podręcznej na hoście w twojej sieci. Oznacza to, że pierwsze zapytanie DNS o określonego hosta jest wysyłane przez łącze szeregowe, ale odpowiedź na każde kolejne będzie wysyłana bezpośrednio z twojego lokalnego serwera nazw i będzie realizowana dużo szybciej. Konfiguracja ta jest opisana w podrozdziale *Konfiguracja named jako serwera pamięci podręcznej* rozdziału 6.

Dla potrzeb naszego przykładu realizacji połączenia PPP z *pppd* załóżmy, że znów jesteś na hoście **vlager**. Najpierw dzwonisz do serwera PPP, **c3po,** i logujesz się na koncie **ppp**. Serwer **c3po** uruchamia swój sterownik PPP. Po zakończeniu pracy z programem komunikacyjnym używanym do dzwonienia, jest wykonywane następujące polecenie, w którym musisz zastąpić pokazaną nazwę urządzenia szeregowego `ttyS3` swoją nazwą.

```
# pppd /dev/ttyS3 38400 crtscts defaultroute
```

Polecenie to przełącza łącze szeregowe *ttyS3* na protokół obsługi PPP i negocjuje łącze IP z **c3po**. Prędkość używana na tym porcie szeregowym wynosi 38 400 bitów na sekundę.

Opcja `crtscts` włącza naporcie uzgadnianie sprzętowe, które jest bezwzględnie wymagane przy prędkościach powyżej 9600 bps.

Po uruchomieniu *pppd* w pierwszej kolejności negocjuje kilka charakterystyk łącza z drugą stroną za pomocą LCP. Zwykle wystarcza domyślny zestaw opcji *pppd*, a więc nie będziemy tu rozwijać tego tematu. Wystarczy powiedzieć, że ta negocjacja częściowo dotyczy żądania lub przypisania adresów IP dla każdej ze stron połączenia.

Na razie zakładamy, że serwer **c3po** nie wymaga od nas żadnego uwierzytelnienia, a więc faza konfiguracji kończy się pełnym sukcesem.

pppd będzie następnie negocjować parametry IP z drugą stroną, używając IPCP – protokołu sterującego IP. Ponieważ wcześniej nie określiliśmy żadnego szczególnego adresu IP dla *pppd*, to będzie on próbować wykorzystać adres uzyskany od resolvera sprawdzającego lokalną nazwę hosta. Następnie obie strony przekażą sobie wzajemnie swoje adresy.

Zwykle w ustawieniach domyślnych nie ma nic złego. Nawet jeżeli twój komputer znajduje się w sieci Ethernet, możesz mieć ten sam adres IP zarówno dla interfejsu Ethernet, jak i PPP. Jednak *pppd* pozwala użyć innego adresu, a nawet zasugerować drugiej stronie użycie jakiegoś określonego adresu. Opcje te omawiamy dalej, w podrozdziale *Opcje konfiguracyjne IP*.

Po przejściu przez fazę konfiguracji IPCP, *pppd* przygotuje warstwę sieciową twojego hosta do działania w roli łącza PPP. Najpierw konfiguruje interfejs sieciowy PPP jako łącze punkt-punkt, używając *ppp0* dla pierwszego aktywnego łącza PPP, *ppp1* dla drugiego i tak dalej. Następnie dokonuje wpisu w tablicy routingu, tak by wskazywał on na hosta po drugiej stronie łącza. W poprzednim przykładzie *pppd* ustawił domyślny routing do sieci na **c3po**, ponieważ podaliśmy go w opcji `defaultroute`*. Obecność trasy domyślnej upraszcza routing, gdyż wszelkie datagramy IP nieprzeznaczone dla hosta lokalnego są wysyłane do **c3po**. Ma to sens, ponieważ jest to jedyna trasa, którą można do niego dotrzeć. Istnieje szereg różnych schematów routingu obsługiwanych przez *pppd*. Omówimy je szczegółowo w dalszej części tego rozdziału.

Używanie plików opcji

Zanim *pppd* dokona analizy składniowej argumentów wiersza poleceń, przegląda kilka plików w poszukiwaniu opcji domyślnych. Pliki te mogą zawierać wszelkie dopuszczalne argumenty wiersza poleceń rozrzucone po wielu wierszach. Znaki hasha (#) oznaczają komentarze.

Pierwszym plikiem opcji jest */etc/ppp/options*. Jest on zawsze przeglądany podczas uruchamiania *pppd*. Użycie tego pliku do ustawienia kilku globalnych wartości do-

* Domyślna trasa do sieci jest instalowana tylko wtedy, gdy żadna inna nie jest do tej pory zdefiniowana.

myślnych jest dobrym pomysłem, ponieważ pozwala powstrzymać użytkowników od zrobienia pewnych rzeczy, które mogą zagrażać bezpieczeństwu systemu. Na przykład, aby włączyć w *pppd* wymóg uwierzytelniania (PAP czy CHAP) dla drugiej strony, wystarczy dodać w tym pliku opcję *auth*. Opcja ta nie może być zmieniona przez użytkownika, a więc niemożliwe staje się zrealizowanie połączenia PPP z jakimkolwiek systemem, który nie znajduje się w autoryzacyjnej bazie danych. Zauważ jednak, że niektóre opcje można zmienić. Dobrym przykładem jest ciąg znaków opcji `connect`.

Inny plik opcji odczytywany po */etc/ppp/options*, to *.ppprc* w katalogu macierzystym użytkownika. Pozwala on każdemu użytkownikowi określić własny zestaw opcji domyślnych.

Przykładowy plik */etc/ppp/options* mógłby wyglądać tak:

```
# Globalne opcje dla pppd dzia□aj□cego na vlager.vbrew.com
lock                # u□yj blokowania urz□dzenia w stylu UUCP
auth                # wymóg uwierzytelnienia
usehostname         # u□yj lokalnej nazwy hosta dla CHAP
domain vbrew.com    # nazwa naszej domeny
```

Słowo kluczowe `lock` powoduje, że *pppd* obsługuje metodę blokowania urządzenia zgodną ze standardem UUCP. W tej konwencji każdy proces, który ma dostęp do urządzenia szeregowego, na przykład */dev/ttyS3*, tworzy w specjalnym katalogu plik blokujący o nazwie postaci *LCK..ttyS3* i w ten sposób informuje, że urządzenie jest używane. Jest to jedyna możliwość, aby inne programy, takie jak *minicom* czy *uucico*, nie otwierały urządzeń szeregowych używanych przez PPP.

Kolejne trzy opcje odnoszą się do uwierzytelniania i co za tym idzie są związane z bezpieczeństwem systemu. Opcje uwierzytelniania najlepiej umieścić w globalnym pliku konfiguracyjnym, ponieważ jest on „uprzywilejowany" i ma wyższy priorytet niż pliki opcji użytkowników *~/.ppprc* (nie mogą oni zmieniać ustawionych w nim opcji).

Stosowanie chat do automatycznego dzwonienia

Jedną z rzeczy w poprzednim przykładzie, która mogła wydać ci się niewygodna, jest to, że musisz ręcznie zrealizować połączenie, zanim będziesz mógł uruchomić *pppd*. W odróżnieniu od *dip*, *pppd* nie ma własnego języka skryptowego pozwalającego na dzwonienie i logowanie się do zdalnych systemów, ale korzysta z zewnętrznego programu lub skryptu powłoki. Polecenie do wykonania może być podane *pppd* za pomocą opcji wiersza poleceń `connect`. *pppd* przekierujestandardowe wejście i wyjście polecenia do łącza szeregowego.

Pakiet oprogramowania *pppd* zawiera bardzo prosty program *chat*, który może być używany do automatyzacji prostych sekwencji logowania. Polecenie to omówimy bardziej szczegółowo.

Jeżeli twoja sekwencja logowania jest złożona, będziesz potrzebował czegoś lepszego niż *chat*. Na pewno wart rozważenia jest *expect*, napisany przez Dona Libesa. Ma bardzo wydajny język oparty na Tcl i został przewidziany właśnie do takich zadań.

Jeśli masz sekwencję logowania, która wymaga na przykład uwierzytelnienia typu wywołanie/odpowiedź, opartego na kalkulatorowych generatorach kluczy, przekonasz się, że *expect* jest wystarczająco dobry, by zrealizować to zadanie. Ponieważ możliwości są tu tak duże, nie będziemy opisywali, jak stworzyć odpowiedni skrypt *expecta*. Dość powiedzieć, że swój skrypt *expect* możesz wywołać, podając jego nazwę w opcji `connect` *pppd*. Trzeba także wiedzieć, że gdy skrypt zostanie uruchomiony, standardowe wejście i wyjście zostaną podłączone do modemu, a nie do terminala, z którego został wywołany *pppd*. Jeżeli wymagana jest interakcja z użytkownikiem, powinieneś ją obsłużyć, otwierając dodatkowy wirtualny terminal lub w jakiś inny sposób.

Polecenie *chat* pozwala ci stworzyć skrypt dialogowy w stylu UUCP. Zasadniczo skrypt *chat* składa się z kolejnych sekwencji ciągów znaków, których oczekujemy od zdalnego systemu, i odpowiedzi, które na nie wysyłamy. Nazywamy je odpowiednio ciągiem *oczekiwanym* (ang. *expect string*) i ciągiem *wysyłanym* (ang. *send string*). Oto typowy fragment skryptu dialogowego:

```
ogin: b1ff ssword: s3|<r1t
```

Ten skrypt informuje *chat*, że by czekał, aż system zdalny przyśle monit logowania, i w odpowiedzi wysłał nazwę użytkownika **b1ff**. Oczekujemy tylko na `ogin:`, a więc nie ma znaczenia, czy monit logowania zaczyna się dużą, czy małą literą l. Następny ciąg powoduje, że *chat* czeka na monit hasła i wysyła w odpowiedzi nasze hasło.

W zasadzie jest to wszystko, co robią skrypty dialogowe. Pełny skrypt dzwoniący do serwera PPP oczywiście musiałby zawierać odpowiednie polecenia modemu. Załóżmy, że twój modem rozumie zestaw poleceń Hayesa, a numer telefonu serwera to 318714. Pełne wywołanie *chat* realizujące połączenie z **c3po** byłoby następujące:

```
$ chat -v '' ATZ OK ATDT318714 CONNECT '' ogin: ppp word: GaGariN
```

Z definicji pierwszy ciąg musi być ciągiem oczekiwanym, ale ponieważ modem nic nie przyśle, zanim go nie zainicjujemy, ustawiliśmy *chat* tak, aby pominął pierwszy oczekiwany ciąg znaków, podając ciąg pusty. Następnie wysyłamy `ATZ` – polecenie zerowania modemów kompatybilnych ze standardem Hayesa i czekamy na odpowiedź (`OK`). Kolejny ciąg znaków wysyła do *chat* polecenie *dial* wraz z numerem telefonu i oczekuje w odpowiedzi komunikatu `CONNECT`. Dalej znów następuje pusty ciąg znaków, ponieważ nie chcemy teraz nic wysyłać, a raczej czekamy na monit logowania. Pozostała część skryptu dialogowego działa dokładnie tak, jak opisaliśmy wcześniej. Opis ten prawdopodobnie wygląda na nieco zagmatwany, ale za chwilę zobaczymy, że istnieje sposób na stworzenie skryptów dialogowych dużo łatwiejszych do zrozumienia.

Opcja *-v* powoduje, że *chat* loguje wszystkie działania przez funkcję `local2`* demona *syslog*.

* Jeżeli dokonasz edycji pliku *syslog.conf* i przekierujesz te komunikaty do pliku, sprawdź, czy plik ten nie jest czytelny dla wszystkich, gdyż *chat* domyślnie wpisuje tam również cały skrypt dialogowy – włącznie z hasłami.

Podanie skryptu dialogowego w wierszu poleceń jest ryzykowne, ponieważ użytkownicy mogą podejrzeć wiersz poleceń za pomocą *ps*. Ryzyka tego możesz uniknąć, umieszczając skrypt dialogowy w pliku na przykład *dial-c3po*. Następnie zmuszasz *chat* do czytania skryptu z pliku zamiast z wiersza poleceń, podając opcję *-f*, a po niej nazwę pliku. Takie podejście ma dodatkową zaletę – ułatwia zrozumienie sekwencji skryptu dialogowego. Po zamianie naszego przykładu na plik *dial-c3po* będzie on wyglądał następująco:

```
''         ATZ
OK         ATDT318714
CONNECT    ''
ogin:      ppp
word:      GaGariN
```

W tej postaci skryptu dialogowego oczekiwany ciąg znaków znajduje się po lewej stronie, a to co wysyłamy w odpowiedzi – po prawej. Coś pokazanego w taki sposób czyta się dużo łatwiej.

Pełne wywołanie *pppd* teraz wyglądałoby następująco:

```
# pppd connect "chat -f dial-c3po" /dev/ttyS3 38400 -detach \
        crtscts modem defaultroute
```

Poza opcją `connect` określającą skrypt, podaliśmy w wierszu poleceń dwie dodatkowe opcje: `-detach`, która mówi *pppd*, by nie odłączał się od konsoli i nie stawał się procesem działającym w tle, oraz słowo kluczowe `modem`, które realizuje działania specyficzne dla modemu widocznego jako urządzenie szeregowe, czyli rozłączenie linii przed dzwonieniem i po nim. Jeżeli nie użyjesz tego słowa kluczowego, *pppd* nie będzie sprawdzało linii DCD portu i nie wykryje, czy przypadkiem druga strona się nie zawiesiła.

Pokazane przykłady są raczej proste. *chat* pozwala na tworzenie dużo bardziej skomplikowanych skryptów. Na przykład można określić ciąg znaków, przy którym dialog zostanie przerwany z błędem. Typowe ciągi przerywające komunikat `BUSY` czy `NO CARRIER`. Modem zwykle je generuje, gdy wywoływany numer jest zajęty albo nie odpowiada. Aby *chat* rozpoznawał te komunikaty natychmiast, możesz je wpisać na początku skryptu, używając słowa kluczowego `ABORT`:

```
$ chat -v ABORT BUSY ABORT 'NO CARRIER' '' ATZ OK ...
```

Podobnie możesz zmienić wartości czasu oczekiwania dla części skryptów dialogowych, wstawiając tam opcje `TIMEOUT`.

Czasami potrzebne jest także warunkowe wykonywanie części skryptu *chat*: gdy nie otrzymasz monitu logowania drugiej strony, zapewne zechcesz wysłać BREAK lub powrót karetki. Możesz to zrobić, dodając podskrypt do oczekiwanego ciągu. Podskrypt składa się z oddzielonych kreską sekwencji ciągów wysyłanego i oczekiwanego, tak jak normalny skrypt. Podskrypt jest wykonywany wtedy, gdy oczekiwany ciąg znaków, do którego jest dołączony, nie nadejdzie na czas. Powyższy przykład moglibyśmy zmodyfikować następująco:

```
ogin:-BREAK-ogin: ppp ssword: GaGariN
```

Gdy *chat* nie zobaczy monitu logowania zdalnego systemu, wywoływany jest podskrypt, który najpierw wysyła BREAK, a następnie czeka ponownie na monit logowania. Jeżeli teraz monit się pojawi, skrypt działa dalej normalnie. W przeciwnym razie kończy działanie z błędem.

Opcje konfiguracyjne IP

Protokół IPCP jest używany do negocjowania szeregu parametrów IP w czasie konfiguracji łącza. Zwykle każda ze stron wysyła pakiet żądania konfiguracji IPCP (ang. *IPCP configuration request*) zawierający zmienne, których wartość domyślną chce zmienić. Po jego otrzymaniu strona zdalna sprawdza każdą opcję po kolei i potwierdza ją albo odrzuca.

pppd daje ci dużą kontrolę nad opcjami IPCP, które próbuje negocjować. Możesz je dostosowywać przez różne opcje wiersza poleceń, które omawiamy poniżej.

Wybór adresów IP

Wszystkim interfejsom trzeba przypisać adresy IP. Urządzenie PPP zawsze ma adres IP. W zestawie protokołów PPP znajduje się mechanizm pozwalający na automatyczne przypisanie adresów IP do interfejsów PPP. Program PPP po jednej stronie łącza punkt-punkt może przypisać adres IP drugiemu końcowi, ale możliwe jest także, by każdy używał własnego adresu IP.

Niektóre serwery PPP obsługujące wielu klientów przypisują adresy dynamicznie. Są one przypisywane do systemów tylko wtedy, gdy te zadzwonią, a odbierane im, gdy się wylogują. Pozwala to na ograniczenie liczby adresów IP do liczby linii komutowanych. Choć ograniczenie to jest wygodne dla zarządców komutowanego serwera PPP, często jest mniej wygodne dla dzwoniących do niego użytkowników. W rozdziale 6 omówiliśmy sposób odzorowywania nazw hostów na adresy IP za pomocą bazy danych. Aby ludzie mogli podłączyć się do twojego hosta, muszą znać jego adres IP lub nazwę. Jeżeli jesteś użytkownikiem usługi PPP, która przypisuje ci adres dynamicznie, trudno będzie ci się tego dowiedzieć bez uzyskania czegoś w rodzaju pozwolenia na uaktualnianie bazy danych DNS po przypisaniu adresu IP. Takie systemy istnieją, ale nie będziemy ich tu szczegółowo omawiać. Przyjrzymy się natomiast preferowanemu podejściu, które polega na użyciu tego samego adresu IP za każdym razem, gdy ustanawiasz swoje połączenie sieciowe*.

W poprzednim przykładzie mieliśmy host **c3po,** na którym działał demon *pppd*, i z nim zestawiliśmy połączenie IP. Nie postawiliśmy warunku, by pojakiejkolwiek stronie połączenia został wybrany konkretny adres IP. Zamiast tego pozwoliliśmy *pppd* na realizację swojego działania domyślnego. Demon ten próbował znaleźć adres IP dla nazwy hosta lokalnego, w naszym przykładzie **vlager**, który wykorzystał po stronie lokalnej oraz pozwolił maszynie zdalnej **c3po** na ustalenie własnego adresu. PPP obsługuje kilka różnych sposobów przypisywania numerów IP.

---

* Więcej informacji na temat dwóch mechanizmów dynamicznego przypisywania hostów znajdziesz pod adresami: *http://www.dynip.com* i *http://www.justlinux.com/dynamic_dns.html*.

Aby uzyskać konkretne adresy, wywołujesz *pppd* z następującą opcją:

`adres_lokalny:adres_zdalny`

`adres_lokalny` i `adres_zdalny` mogą być podane zarówno w postaci liczbowej, jak i w postaci nazw hostów*. Ta opcja powoduje, że *pppd* próbuje używać pierwszego z dostarczonych adresów jako własnego adresu IP, a drugiego jako adresu partnera. Jeżeli partner odmówi przyjęcia któregokolwiek adresu w czasie negocjacji IPCP, łącze nie zostanie uaktywnione**.

Jeżeli dzwonisz do serwera po adres IP, powinieneś sprawdzić, czy *pppd* nie próbuje wynegocjować adresu dla siebie. W tym celu użyj opcji *noipdefault* i pozostaw pole `adres_lokalny` puste. Opcja *noipdefault* powstrzymademona *pppd* przed próbą użycia adresu IP związanego z nazwą hosta jako adresu lokalnego.

Gdybyś chciał ustawić tylko adres lokalny oraz przyjąć adres używany przez partnera, po prostu pozostaw puste pole `adres_zdalny`. Aby **vlager** używał adresu IP **130.83.4.27** zamiast swojego własnego, podaj w wierszu poleceń *130.83.4.27:*. Podobnie, aby ustawić jedynie adres zdalny, pozostaw puste pole `adres_lokalny`. Domyślnie *pppd* używa adresu związanego z nazwą twojego hosta.

Ruting przez łącze PPP

Po skonfigurowaniu interfejsu sieciowego, *pppd* zwykle konfiguruje jedynie trasę hosta do jego partnera. Jeżeli zdalny host jest w sieci LAN, pewnie chcesz łączyć się także z hostami „poza" twoim partnerem. W takiej sytuacji należy skonfigurować także trasę dla sieci.

Widzieliśmy już, że za pomocą opcji *defaultroute*, można poprosić *pppd* o skonfigurowanie trasy domyślnej. Opcja ta jest bardzo przydatna, jeżeli serwer PPP, do którego zadzwoniłeś, działa jako twój gateway internetowy.

Odwrotny przypadek, w którym twój system działa jako gateway dla pojedynczego hosta, jest także stosunkowo łatwo zrealizować. Na przykład weźmy pracownika browaru wirtualnego, którego maszyna domowa nazywa się **oneshot**. Załóżmy także, że skonfigurowaliśmy **vlager** jako wdzwaniany serwer PPP. Jeżeli skonfigurowaliśmy **vlager** do dynamicznego przypisywania adresu IP należącego do podsieci browaru, możemy użyć w *pppd* opcji *proxyarp*, która zainstaluje wpis proxy ARP dla **oneshota**. Automatycznie **oneshot** stanie się dostępny ze wszystkich hostów w browarze i winiarni.

Jednak nie zawsze jest to takie proste. Połączenie dwóch sieci lokalnych zwykle wymaga dodania szczególnego rutingu do sieci, ponieważ sieci te mogą mieć własne trasy domyślne. Poza tym, gdyby obie strony łącza PPP były domyślnymi trasami dla każdej z sieci, mogłaby powstać pętla, w której wszystkie pakiety o nieznanym

* Użycie nazw hostów w tej opcji ma wpływ na uwierzytelnianie CHAP. Zajrzyj do podrozdziału *Uwierzytelnianie w PPP* w dalszej części tego rozdziału.

** Opcje *ipcp-accept-local* i *ipcp-accept-remote* mówią twojemu demonowi *pppd*, aby zaakceptował lokalny i zdalny adres IP oferowany przez zdalne PPP, nawet jeżeli podałeś jakieś adresy w swojej konfiguracji. Jeżeli te opcje nie są skonfigurowane, twój demon *pppd* odrzuci wszelkie próby negocjacji adresów IP.

przeznaczeniu odbijałyby się pomiędzy końcami łącza PPP, aż do wygaśnięcia ich czasu życia.

Załóżmy, że browar wirtualny otwiera oddział w innym mieście. Ten oddział ma własną sieć Ethernet o adresie IP **172.16.3.0**, która jest trzecią podsiecią sieci klasy B browaru. Filia chce podłączyć się do sieci browaru przez PPP w celu uaktualniania baz klientów. Znów **vlager** działa jako gateway dla sieci browaru i obsługuje łącze PPP. Jego drugi koniec w oddziale nazywa się **vbourbon** i ma adres IP **172.16.3.1**. Sieć ta jest pokazana na rysunku A-2 w dodatku A, *Przykładowa sieć: browar wirtualny*.

Gdy **vbourbon** łączy się z **vlagerem**, ustawia trasę domyślną tak, by wskazywała jak zwykle na **vlager**. Jednak na **vlagerze** będziemy mieli tylko trasę punkt-punkt do **vbourbon** i będziemy musieli specjalnie skonfigurować ruting sieci dla podsieci 3 wykorzystujący **vbourbon** jako gateway. Możemy to zrobić ręcznie, używając polecenia *route* po zestawieniu połączenia PPP, ale nie jest to zbyt praktyczne rozwiązanie. Na szczęście możemy skonfigurować trasę automatycznie, używając funkcji *pppd*, której jeszcze nie omawialiśmy: polecenia *ip-up*. Polecenie to jest skryptem powłoki albo programem ulokowanym w katalogu */etc/ppp* i jest uruchamiane przez *pppd* po skonfigurowaniu interfejsu PPP. Jest wywoływane z następującymi parametrami:

```
ip-up interfejs urz□dzenie pr□dko□□ adr_lok adr_zdal
```

Poniższa tabela podaje znaczenie każdego argumentu (w pierwszej kolumnie pokazujemy liczbę używaną przez skrypt powłoki przy odwoływaniu się do każdego z argumentów):

Argument	*Nazwa*	*Przeznaczenie*
$1	`interfejs`	Używany interfejs sieciowy, np. `ppp0`.
$2	`urz□dzenie`	Ścieżka do pliku używanego urządzenia szeregowego (*/dev/tty* jeżeli jest używane stdin/stdout).
$3	`pr□dko□□`	Prędkość urządzenia szeregowego w b/s.
$4	`adr_lok`	Adres IP lokalnego końca łącza w postaci liczbowej.
$5	`adr_zdal`	Adres IP zdalnego końca łącza w postaci liczbowej.

W naszym przypadku skrypt *ip-up* może zawierać następujący fragment kodu*:

```
#!/bin/sh
case $5 in
172.16.3.1)      # to jest vbourbon
      route add -net 172.16.3.0 gw 172.16.3.1;;
...
esac
exit 0
```

* Gdybyśmy chcieli mieć ruting do innych ośrodków stworzonych przy ich wdzwanianiu się, dodalibyśmy w przykładzie odpowiednie dyrektywy case w miejscach wykropkowanych.

Podobnie */etc/ppp/ip-down* może być użyte do cofnięcia wszelkich działań *ip-up* po rozłączeniu łącza PPP. Tak więc w naszym skrypcie */etc/ppp/ip-down* moglibyśmy mieć polecenie *route* usuwające trasę stworzoną w skrypcie */etc/ppp/ip-up*.

Jednak nie jest to jeszcze pełny schemat rutingu. Stworzyliśmy wpisy w tablicy rutingu na obu hostach PPP, ale do tej pory żaden host w żadnej z tych sieci nic nie wie o łączu PPP. Nie stanowi to problemu, jeżeli wszystkie hosty w oddziale mają własny ruting domyślny wskazujący na **vbourbon**, a wszystkie hosty w browarze mają domyślny ruting na **vlager**. Jeżeli jednak w twojej sytuacji nie jest to dobre rozwiązanie, jedną możliwością jest zwykle zastosowanie demona rutingu, na przykład *gated*. Po utworzeniu trasy do **vlagera**, demon rutingu rozgłasza ją wszystkim hostom w podłączonych pod sieciach.

Opcje sterowania łączem

Spotkaliśmy się już z protokołem sterowania łączem (LCP), który jest używany do negocjacji charakterystyk i testowania łącza.

Dwie najważniejsze opcje negocjowane przez LCP to: *mapa znaków sterujących w transmisji asynchronicznej* (*Asynchronous Control Character Map*) i *maksymalna jednostka odbioru* (*Maximum Receive Unit*). Istnieje szereg innych opcji konfiguracyjnych LCP, ale są one zbyt specjalistyczne, by je tu omawiać.

Mapa znaków sterujących w transmisji asynchronicznej, potocznie zwana *mapą asynchroniczną* (ang. *async map*), jest używana w łączach asynchronicznych, takich jak linie telefoniczne, do identyfikowania znaków sterujących, które muszą być maskowane (zastępowane przez specyficzną sekwencję dwuznakową), aby nie zostały zinterpretowane przez urządzenia używane do zestawiania połączenia. Na przykład można w ten sposób uniknąć znaków XON i XOFF używanych przy programowym uzgadnianiu, ponieważ źle skonfigurowane modemy mogłyby się zawiesić po otrzymaniu XOFF. Inny potencjalnie niebezpieczny znak to [CTRL+l] (znak ucieczki *telnet*). PPP pozwala na maskowanie wszelkich znaków o kodach ASCII od 0 do 31 przez umieszczenie ich w mapie asynchronicznej.

Mapa asynchroniczna jest 32-bitowym ciągiem wyrażonym w postaci liczby szesnastkowej. Najmniej znaczący bit odpowiada znakowi NULL ASCII, a najbardziej znaczący bit odpowiada znakowi o dziesiętnym kodzie 31 w zestawie ASCII. Te 32 znaki ASCII są znakami sterującymi. Jeżeli bit w ciągu jest ustawiony, sygnalizuje, że odpowiadający mu znak musi być zamaskowany przed przesłaniem przez łącze.

Aby powiadomić partnera, że nie musi maskować wszystkich znaków sterujących, a tylko kilka, możesz wprowadzić mapę asynchroniczną do *pppd* za pomocą opcji *asyncmap*. Na przykład jeżeli muszą być maskowane tylko znaki `^S` i `^Q` (ASCII 17 i 19, powszechnie używane dla XON i XOFF), użyj poniższej opcji:

```
asyncmap 0x000A0000
```

Konwersja jest prosta, gdyż polega jedynie na zamianie liczby binarnej na szesnastkową. Narysuj sobie 32 bity. Skrajny bit z prawej strony odpowiada znakowi ASCII 00 (NULL), a z lewej strony znakowi ASCII 32 (dziesiętnie). Ustaw bity odpowia-

dające znakom, które chcesz maskować, a wszystkie inne pozostaw wyzerowane. Aby zamienić ten ciąg na liczbę szesnastkową, której oczekuje *pppd*, po prostu weź każde 4 bity i wyraź je w postaci liczby szesnastkowej. Powinieneś uzyskać osiem cyfr szesnastkowych. Ułóż z nich ciąg i poprzedź znakami „0x”, aby wskazać, że jest to liczba szesnastkowa. I gotowe.

Początkowo mapa asynchroniczna jest ustawiona na `0xffffffff`, czyli wszystkie znaki sterujące są zamaskowane. Jest to bezpieczna wartość domyślna, ale zwykle jest to dużo więcej, niż potrzebujesz. Każdy znak, który znajduje się w mapie asynchronicznej, daje dwa znaki w czasie przesyłania przez łącze, a więc maskowanie jest realizowane kosztem zwiększonego wykorzystania łącza i zmniejszenia wydajności.

Zwykle dobrze działa mapa asynchroniczna o wartości `0x0`. W takim przypadku żadne maskowanie nie jest realizowane.

Maksymalna jednostka odbioru (MRU) przekazuje partnerowi informację o maksymalnym rozmiarze ramek HDLC, które chcemy odbierać. Choć MRU może ci się kojarzyć z maksymalną jednostką transmisji (MTU), ma z nią jednak niewiele wspólnego. MTU to parametr urządzenia sieciowego jądra i opisuje maksymalny rozmiar ramki, którą jest w stanie wysłać interfejs. MRU to informacja dla zdalnego końca, mówiąca mu, aby nie generował ramek większych niż MRU. Mimo to interfejs musi mieć możliwość odbierania ramek o wielkości do 1500 bajtów.

Wybór MRU nie jest więc kwestią tego, co łącze może przenieść, ale raczej tego, co daje najlepszą przepustowość. Jeżeli zamierzasz korzystać z aplikacji interaktywnych, ustawienie MRU na wartości tak małe jak 296 bajtów jest dobrym pomysłem, a sporadyczne większe pakiety (powiedzmy sesji FTP) nie spowodują, że twój kursor będzie „skakał”. Aby *pppd* żądało MRU o wartości 296, musisz podać opcję `mru 296`. Jednak małe MRU ma sens tylko, jeżeli masz kompresję nagłówków VJ (jest ona domyślnie włączona), ponieważ w przeciwnym razie tracisz sporą część łącza na przesyłanie nagłówka IP każdego datagramu.

pppd rozumie także kilka opcji LCP konfigurujących ogólne zachowanie procesu negocjacji, takich jak maksymalna liczba żądań konfiguracyjnych, które mogą być wymienione przed rozłączeniem łącza. Dopóki dokładnie nie wiesz, co robisz, pozostaw te opcje w spokoju.

I na koniec, istnieją dwie opcje, które dotyczą powtarzania komunikatów LCP. PPP definiuje dwa komunikaty *Echo Request* i *Echo Response*. *pppd* używa tej funkcji do sprawdzania, czy łącze wciąż działa. Możesz ją włączyć za pomocą opcji *lcp-echo-interval*, podając czas w sekundach. Jeżeli w zadanym przedziale czasu z hosta zdalnego nie zostaną odebrane żadne ramki, *pppd* generuje Echo Request i będzie oczekiwał aż partner zwróci Echo Response. Jeżeli partner nie odpowie, połączenie jest przerywane po wysłaniu pewnej liczby żądań. Liczba ta może być ustalona za pomocą opcji *icp-echo-failure*. Domyślnie funkcja ta jest wyłączona.

Uwagi na temat bezpieczeństwa

· le skonfigurowany demon PPP może stanowić poważne zagrożenie. Może pozwolić każdemu na podłączenie swojego komputera do twojej sieci Ethernet (co jest bardzo niebezpiecznie). W tym podrozdziale omówimy kilka środków zaradczych, dzięki którym konfiguracja twojego PPP będzie bezpieczna.



Do skonfigurowania urządzenia sieciowego i tablicy rutingu są potrzebne uprawnienia roota. Zwykle rozwiązuje się ten problem, uruchamiając *pppd* z prawem setuid **root**. Jednak *pppd* pozwala użytkownikom na ustawianie różnych opcji mających wpływ na bezpieczeństwo.

Aby się zabezpieczyć przed atakami, na które może narazić nas użytkownik grzebiący w opcjach demona *pppd*, powinieneś ustawić kilka domyślnych wartości w pliku globalnym */etc/ppp/options*, na przykład w sposób pokazany w przykładowym pliku we wcześniejszym podrozdziale *Używanie plików opcji*. Niektóre z nich, takie jak opcje uwierzytelniania, nie mogą być zmienione przez użytkownika i dzięki temu dają sensowne zabezpieczenie przed manipulacjami. Ważną opcją zabezpieczającą jest `connect`. Jeżeli masz zamiar pozwolić użytkownikom nie mającym uprawnień roota na wywoływanie *pppd* i łączenie się z Internetem, powinieneś zawsze dodać opcje `connect` i `noauth` w globalnym pliku opcji */etc/ppp/options*. Jeżeli tego nie zrobisz, użytkownicy będą mogli uruchamiać różne polecenia z prawami użytkownika root, podając je jako polecenia `connect` w wierszu poleceń *pppd* albo umieszczając w swoim prywatnym pliku opcji.

Innym dobrym pomysłem jest ograniczenie liczby użytkowników, którym wolno uruchamiać *pppd*. W tym celu należy utworzyć grupę w pliku */etc/group* i dodać do niej tylko tych, którzy mogą uruchamiać demona PPP. Następnie trzeba zmienić prawa dodemona *pppd*, tak aby miała do niego dostęp ta grupa i usunąć prawo uruchamiania dla pozostałych osób. Zakładając, że nazwałeś swoją grupę **dialout**, możesz zrobić coś takiego:

```
# chown root /usr/sbin/pppd
# chgrp dialout /usr/sbin/pppd
# chmod 4750 /usr/sbin/pppd
```

Oczywiście musisz się zabezpieczyć także przed systemami, z którymi łączysz się przez PPP. Aby obronić się przed hostami udającymi kogoś innego, powinieneś zawsze wymagać od drugiej strony jakiegoś uwierzytelnienia. Nie powinieneś pozwalać obcym hostom na używanie wybranych przez nie adresów IP. Należy im natomiast wskazać kilka dogodnych dla ciebie adresów. Następny podrozdział szczegółowo opisuje te tematy.

Uwierzytelnianie w PPP

W PPP każdy system może zażądać uwierzytelnienia partnera za pomocą jednego z dwóch protokołów uwierzytelniających: *protokołu uwierzytelniania hasłem* (*Password Authentication Protocol* – PAP) i *protokołu uwierzytelnienia przez uzgodnienie* (*Challenge*

Handshake Authentication Protocol – CHAP). Gdy połączenie zostanie zestawione, każda strona może zażądać od drugiej uwierzytelnienia się, bez względu na to, czy jest stroną wywołującą, czy wywoływaną. W dalszym opisie będziemy luźno mówili o „kliencie" i „serwerze", gdy będziemy chcieli rozróżnić system wysyłający żądanie uwierzytelnienia od systemu na nie odpowiadającego. Demon PPP może zażądać uwierzytelnienia partnera, wysyłając żądanie konfiguracyjne LCP identyfikujące wybrany protokół uwierzytelniania.

PAP a CHAP

PAP, oferowany przez wielu usługodawców internetowych, działa w zasadzie w ten sam sposób jak normalna procedura logowania. Klient uwierzytelnia się, wysyłając nazwę użytkownika i (opcjonalnie zaszyfrowane) hasło do serwera, który porównuje je z bazą danych sekretów*. Ta technika nie stanowi zabezpieczenia przed podsłuchiwaczami, którzy mogą spróbować uzyskać hasło, słuchając danych przesyłanych przez łącze szeregowe, i atakować metodą prób i błędów.

CHAP nie ma tych niedostatków. W przypadku CHAP serwer wysyła losowo wygenerowany ciąg „wywołania" do klienta wraz ze swoją nazwą hosta. Klient wykorzystuje nazwę hosta do wyszukania odpowiedniego sekretu, łączy go z wywołaniem i szyfruje ciąg za pomocą jednokierunkowej funkcji mieszającej. Wynik jest zwracany do serwera wraz z nazwą hosta klienta. Serwer teraz wykonuje te same obliczenia i potwierdza wiarygodność klienta, jeżeli uzyska ten sam wynik.

CHAP również nie wymaga, by klient sam uwierzytelniał się tylko na początku, ale wysyła wywołania w regularnych odstępach czasu w celu sprawdzenia, czy za klienta nie został podstawiony ktoś niepożądany, na przykład przez przełączenie linii telefonicznych, lub czy nie wystąpił błąd konfiguracji modemu, który spowodował, że demon PPP nie zauważył, że oryginalne połączenie zostało zerwane, a ktoś inny wdzwonił się na to miejsce.

pppd przechowuje sekrety dla PAP i CHAP w dwóch oddzielnych plikach */etc/ppp/pap-secrets* i */etc/ppp/chap-secrets*. Wpisując zdalnego hosta w jednym lub drugim z nich, kontrolujesz, który z tych protokołów (PAP czy CHAP) jest używany do uwierzytelniania się u twojego partnera i odwrotnie.

Domyślnie *pppd* nie wymaga uwierzytelniania zdalnego hosta, ale zgodzi się sam uwierzytelnić, gdy zażąda tego zdalny host. Ponieważ CHAP jest dużo silniejszym protokołem niż PAP, *pppd* próbuje zawsze go używać, o ile to jest tylko możliwe. Jeżeli druga strona nie obsługuje CHAP, albo jeżeli *pppd* nie może znaleźć w pliku *chap-secrets* sekretu CHAP dla zdalnego systemu, przełącza się na PAP. Jeżeli nie istnieje sekret PAP dla drugiej strony, *pppd* w ogóle odmawia uwierzytelnienia. W konsekwencji połączenie jest zrywane.

Zachowanie to możesz zmienić na kilka sposobów. Gdy podasz słowo kluczowe *auth*, *pppd* zażąda, by druga strona sama się uwierzytelniła. *pppd* zgadza się użyć

* „Sekret" to po prostu określenie hasła stosowane w PPP. W odróżnieniu od haseł w Linuksie, sekretów PPP nie obowiązuje ograniczenie długości.

PAP lub CHAP, dopóki posiada w bazie danych odpowiednie sekrety drugiej strony. Istnieją inne opcje pozwalające na włączenie lub wyłączenie zadanego protokołu uwierzytelniania, ale nie będziemy ich tu taj opisywać.

Gdyby wszystkie systemy, z którymi łączysz się przez PPP, zgadzały się same uwierzytelniać, powinieneś umieścić opcję *auth* w globalnym pliku */etc/ppp/options* i zdefiniować hasła dla każdego z tych systemów w pliku *chap-secrets*. Jeżeli system nie obsługuje CHAP, dodaj dla niego wpis w pliku *pap-secrets*. Dzięki temu nieautoryzowane systemy nie podłączą się do twojego hosta.

Dwa następne podrozdziały omawiają dwa pliki sekretów PPP: *pap-secrets* i *chap-secrets*. Znajdują się one w katalogu */etc/ppp* i zawierają trójki klient, serwer i hasło, po których opcjonalnie następuje lista adresów IP. Interpretacja pól klienta i serwera jest różna dla CHAP i PAP i zależy od tego, czy sami się uwierzytelniamy u partnera, czy żądamy, aby serwer uwierzytelnił się u nas.

Plik sekretów CHAP

Gdy *pppd* musi się uwierzytelnić na serwerze za pomocą CHAP, przeszukuje plik *chap-secrets* w poszukiwaniu wpisu, w którym pole klienta jest takie samo jak lokalna nazwa hosta, a pole serwera jest takie samo jak nazwa hosta zdalnego wysłana w wywołaniu CHAP. Gdy wymagane jest samodzielne uwierzytelnienie się partnera, role po prostu się odwracają: *pppd* wtedy szuka wpisu, w którym pole klienta jest takie samo jak nazwa hosta zdalnego (wysłana w odpowiedzi CHAP klienta), a pole serwera jest takie samo jak nazwa hosta lokalnego.

Poniżej pokazano przykładowy plik *chap-secrets* dla hosta **vlager***.

```
# Sekrety CHAP dla vlager.vbrew.com
#
# klient           serwer           sekret                   adresy
#------------------------------------------------------------------------
vlager.vbrew.com   c3po.lucas.com   "Use The Source Luke"    vlager.vbrew.com
c3po.lucas.com     vlager.vbrew.com "arttoo! arttoo!"        c3po.lucas.com
*                  vlager.vbrew.com "TuXdrinksVicBitter"     pub.vbrew.com
```

Gdy **vlager** zestawi połączenie PPP z **c3po**, ten poprosi **vlagera** o uwierzytelnienie się przez wysłanie wywołania CHAP. *pppd* na hoście **vlager** sprawdzi następnie plik *chap-secrets* w poszukiwaniu wpisu, w którym pole klienta ma wartość **vlager.vbrew.com**, a pole serwera ma wartość **c3po.lucas.com**, i znajdzie pierwszy wiersz pokazany w przykładzie. Następnie na podstawie ciągu wywołania generuje odpowiedź i sekret CHAP (`Use The Source Luke`) i wysyła je do **c3po**.

pppd tworzy także wywołanie CHAP dla **c3po**, zawierające unikatowy ciąg wywołania i pełną nazwę domenową hosta, **vlager.vbrew.com**. Host **c3po** tworzy odpowiedź CHAP w omówiony sposób i zwraca ją do **vlagera**. *pppd* następnie wydobywa nazwę hosta klienta (**c3po.vbrew.com**) z odpowiedzi i przeszukuje plik *chap-secrets* w celu znalezienia wiersza zawierającego klienta **c3po** i serwer **vlager**.

* Podwójne cudzysłowy nie są częścią sekretu – mają ułatwić poprawną interpretację białych znaków.

Drugi wiersz pasuje, a więc *pppd* łączy wywołanie CHAP i sekret `arttoo! arttoo!`, szyfruje je i porównuje wynik z odpowiedzią CHAP klienta **c3po**.

Czwarte pole opcjonalne zawiera adresy IP, które są dopuszczalne dla klienta o nazwie zawartej w pierwszym polu. Adresy mogą być podane w zapisie liczbowym lub jako nazwy hostów, które są następnie rozwiązywane przez resolver. Na przykład, gdyby w czasie negocjacji IPCP, klient **c3po** zażądał użycia adresu IP, którego nie ma na liście, żądanie zostałoby odrzucone, a sesja IPCP zakończona. Dlatego w pokazanym powyżej przykładowym pliku **c3po** może używać własnego adresu IP. Gdyby pole adresu było puste, dopuszczalne byłyby wszystkie adresy, a wartość „-" zapobiegałaby w ogóle użyciu adresu IP w przypadku tego klienta.

Trzeci wiersz w przykładowym pliku *chap-secrets* pozwala, aby dowolny host stworzył połączenie PPP z **vlagerem**, ponieważ pole klienta lub serwera zawiera znak *, który pasuje do dowolnej nazwy hosta. Jedynym wymogiem jest to, że podłączający się host musi znać sekret i używać adresu IP związanego z hostem **pub.vbrew.com**. Wpisy z wyrażeniami regularnymi w nazwie hosta mogą pojawić się w dowolnym miejscu pliku sekretów, ponieważ *pppd* zawsze będzie używać tego, co najlepiej pasuje do pary serwer-klient.

pppd może potrzebować nieco pomocy przy tworzeniu nazw hostów. Jak wcześniej wyjaśniliśmy, nazwa hosta zdalnego jest zawsze dostarczana przez drugą stronę w wywołaniu CHAP lub w pakiecie z odpowiedzią. Nazwa hosta lokalnego jest uzyskiwana przez domyślne wywołanie funkcji *gethostname(2)*. Gdybyś ustawił nazwę systemu na niepełną nazwę hosta, musiałbyś dostarczyć *pppd* także nazwę domeny, używając opcji *domain*:

```
# pppd ... domain vbrew.com
```

Ta klauzula dodaje nazwę domeny browaru do **vlagera** w przypadku wszelkich działań związanych z uwierzytelnianiem. Inne opcje modyfikujące pojęcie nazwy hosta lokalnego *pppd* to *usehostname* i *name*. Gdy w wierszu poleceń podasz lokalny adres IP, używając `lokalny:zdalny` i `lokalny` w postaci nazw, a nie liczb, *pppd* uzna je za nazwę hosta lokalnego.

Plik sekretów PAP

Plik sekretów PAP jest bardzo podobny do pliku CHAP. Pierwsze dwa pola zawsze zawierają nazwę użytkownika i nazwę serwera. Trzecie pole zawiera sekret PAP. Gdy zdalny host wysyła swoje informacje uwierzytelniające, *pppd* wykorzystuje wpis, w którym pole serwera odpowiada nazwie lokalnego hosta, a pole użytkownika odpowiada nazwie użytkownika wysłanej w żądaniu. Gdy musimy wysłać swoje referencje do partnera, *pppd* wykorzystuje sekret z wiersza, w którym pole użytkownika odpowiada nazwie lokalnego hosta, a pole serwera nazwie hosta zdalnego.

* Nazwa hosta jest wzięta z wywołania CHAP.

Przykładowy plik sekretów PAP może wyglądać następująco:

```
# /etc/ppp/pap-secrets
#
# u□ytkownik   serwer    sekret         adresy
vlager-pap     c3po      cresspahl      vlager.vbrew.com
c3po           vlager    DonaldGNUth    c3po.lucas.com
```

Pierwszy wiersz jest używany do uwierzytelnienia się przy komunikacji z **c3po**. Drugi opisuje, jak użytkownik **c3po** ma się uwierzytelnić u nas.

Nazwa `vlager-pap` w pierwszej kolumnie to nazwa użytkownika, którą wysyłamy do **c3po**. Domyślnie jako nazwę użytkownika *pppd* przyjmuje nazwę hosta lokalnego, ale możesz podać także inną nazwę, wpisując opcję *user*, a za nią nazwę.

Przy wybieraniu wpisu z pliku *pap-secrets* w celu zidentyfikowania nas na hoście zdalnym, *pppd* musi znać nazwę hosta zdalnego. Ponieważ *pppd* samo nie ma możliwości się tego dowiedzieć, musisz wpisać ją w wierszu poleceń, używając słowa kluczowego *remotename*, a po nim nazwy hosta. Aby za pomocą powyższego wpisu na przykład uwierzytelnić się na **c3po**, musimy dodać poniższą opcję do wiersza poleceń *pppd*:

```
# pppd ... remotename c3po user vlager-pap
```

W czwartym polu pliku sekretów PAP (i wszystkich kolejnych polach) możesz wpisać adresy IP, które mają prawo komunikować się z danym hostem, podobnie jak w pliku sekretów CHAP. Partner będzie miał prawo żądać tylko adresów z tej listy. W przykładowym pliku wpis, którego **c3po** używa, gdy się wdzwania – wiersz, gdzie **c3po** jest klientem – pozwala na użycie jego rzeczywistego adresu IP i żadnego innego.

Zauważ, że PAP jest raczej słabą metodą uwierzytelniania i powinieneś używać CHAP, gdzie to tylko możliwe. Dlatego nie będziemy dokładniej omawiali PAP: jeżeli chcesz go używać, więcej na temat jego funkcji znajdziesz na stronach podręcznika elektronicznego *pppd(8)*.

Debugowanie twojej konfiguracji PPP

Domyślnie *pppd* zapisuje wszelkie ostrzeżenia i błędy za pomocą funkcji `daemon` programu *syslog*. Do pliku *syslog.conf* musisz dodać wpis, który przekierowuje komunikaty do pliku lub na konsolę. W przeciwnym razie *syslog* będzie je po prostu ignorował. Poniżej pokazany wpis powoduje wysyłanie wszystkich komunikatów do pliku */var/log/ppp-log*:

```
daemon.*       /var/log/ppp-log
```

Jeżeli twoja konfiguracja PPP nie działa poprawnie, powinieneś zajrzeć do tego pliku. Jeżeli zawarte w nim komunikaty nie pomogą, możesz włączyć dodatkowe debugowanie, używając opcji *debug*. Spowoduje to, że *pppd* będzie zapisywać zawartość wszystkich pakietów sterujących, wysłanych lub odebranych przez *syslog*. Wszystkie komunikaty będą następnie przekierowywane do funkcji `daemon`.

I na koniec najbardziej drastyczny sposób na poradzenie sobie z problemem, czyli włączenie debugowania na poziomie jądra – robi się to, wywołując *pppd* z opcją *kdebug*. Za nią wpisujemy argument liczbowy będący sumą następujących wartości: 1 – ogólne komunikaty debugujące, 2 – wypisywanie zawartości wszystkich przychodzących ramek HDLC i 4 – sterownik wypisuje wszystkie wychodzące ramki HDLC. Aby przechwycić komunikaty debugujące jądra, musisz uruchomić demona *syslogd*, który czyta plik */proc/kmsg* albo demona *klogd*. Oba te sposoby powodują przekierowanie komunikatów debugujących do funkcji `kernel` demona *syslog*.

Bardziej zaawansowana konfiguracja PPP

Choć konfigurowanie PPP tak, by dzwonić do sieci Internet, jest najpowszechniejszym zastosowaniem, są wśród was tacy, którzy mają bardziej zaawansowane wymagania. W tym podrozdziale omówimy kilka zaawansowanych konfiguracji możliwych do uzyskania w PPP w Linuksie.

Serwer PPP

Uruchomienie *pppd* jako serwera jest jedynie kwestią skonfigurowania urządzenia szeregowego tty na wywoływanie *pppd* z odpowiednimi opcjami, gdy zostaną odebrane przychodzące dane. Aby przeprowadzić taką konfigurację, można utworzyć specjalne konto, powiedzmy **ppp**, i jako powłokę logowania podać program wywołujący *pppd* z tymi opcjami. Alternatywnie, jeżeli chcesz korzystać z uwierzytelniania PAP lub CHAP, możesz użyć programu *mgetty* do obsługi swojego modemu i wykorzystać jego funkcję „/AutoPPP/”.

Aby stworzyć serwer wykorzystujący metodę logowania, do pliku */etc/passwd* musisz dodać wiersz podobny do pokazanego poniżej*:

```
ppp:x:500:200:Public PPP Account:/tmp:/etc/ppp/ppplogin
```

Jeżeli twój system obsługuje system haseł shadow, musisz dodać także wpis do pliku */etc/shadow*:

```
ppp:!:10913:0:99999:7:::
```

Oczywiście użyte przez ciebie identyfikatory UID i GID zależą od tego, który użytkownik ma być właścicielem połączenia i jak je utworzyłeś. Za pomocą polecenia *passwd* musisz także nadać hasło wspomnianemu kontu.

Skrypt *ppplogin* mógłby wyglądać tak:

```
#!/bin/sh
# ppplogin - skrypt uruchamiaj□cy pppd po zalogowaniu
mesg n
stty -echo
exec pppd -detach silent modem crtscts
```

Polecenie *mesg* wyłącza innym użytkownikom możliwość zapisu do tty, na przykład za pomocą polecenia *write*. Polecenie *stty* wyłącza powtarzanie znaków. Jest ono nie-

* Narzędzia *useradd* lub *adduser*, o ile je posiadasz, ułatwią wykonanie zadania.

zbędne, gdyż w przeciwnym razie wszystko, co wyśle druga strona, będzie powtarzane. Najważniejszą opcją podaną w *pppd* jest `-detach`, ponieważ zapobiega odłączeniu *pppd* od kontrolującego go tty. Gdybyśmy nie podali tej opcji, program przeszedłby do pracy w tle, powodując zakończenie skryptu powłoki. Na skutek tego nastąpiłoby rozłączenie linii i utrata połączenia. Opcja *silent* sprawia, że przed rozpoczęciem wysyłania, *pppd* czeka na odebranie pakietu od dzwoniącego systemu. Ta opcja nie pozwala też na pojawienie się w transmisji czasów oczekiwania, jeśli system dzwoniący jest zbyt wolny przy uruchamianiu klienta PPP. Opcja `modem` powoduje, że *pppd* steruje liniami kontrolnymi modemu podłączonego do portu szeregowego. Zawsze powinieneś włączać tę opcję, gdy używasz *pppd* z modemem. Opcja *crtscts* włącza uzgadnianie sprzętowe.

Oprócz wymienionych, są też jeszcze opcje o innym działaniu. Na przykład podając *auth* w wierszu wywołania *pppd* lub w globalnym pliku opcji, możesz wymusić jakieś uwierzytelnienie. Strona podręcznika elektronicznego omawia bardziej szczegółowe opcje włączania i wyłączania poszczególnych protokołów uwierzytelniania.

Gdybyś chciał używać demona *mgetty*, trzeba jedynie skonfigurować go tak, aby obsługiwał urządzenie szeregowe, do którego podłączony jest modem (szczegóły znajdziesz w podrozdziale *Konfigurowanie demona mgetty* w rozdziale 4), skonfigurować *pppd* na uwierzytelnianie przez PAP lub CHAP za pomocą odpowiednich opcji w pliku *options* i wreszcie dodać do pliku */etc/mgetty/login.config* coś takiego:

```
# Konfigurowanie mgetty do automatycznego wykrywania
# przychodz□cych wywo□a□ PPP i uruchomienie demona pppd do
# obs□ugi po□□czenia
#
/AutoPPP/ -   ppp   /usr/sbin/pppd auth -chap +pap login
```

Pierwsze pole to takie magiczne zaklęcie używane do wykrywania, czy nadchodzące połączenie jest typu PPP. Nie możesz zmieniać pisowni tego ciągu, gdyż istotne są w nim duże i małe litery. Trzecia kolumna to nazwa użytkownika, który pojawia się na liście *who*, gdy ktoś się zaloguje. Pozostała część wiersza to polecenie do wywołania. Za pomocą takiego zapisu jak w naszym przykładzie włączamy uwierzytelnianie PAP, wyłączamy CHAP i mówimy, że plik *passwd* powinien być użyty do znalezienia użytkowników do uwierzytelnienia. Zapewne coś podobnego chcesz uzyskać. Pamiętaj – możesz określić opcje w pliku *options* lub jeżeli wolisz, w wierszu poleceń.

Oto krótka lista kolejnych zadań do wykonania, jeżeli chcesz uruchomić na swoim komputerze wdzwaniany serwer PPP. Sprawdź, czy każdy krok został poprawnie zrealizowany, zanim przejdziesz do następnego:

1. Skonfigurowanie modemu do trybu automatycznego odpowiadania. W modemach kompatybilnych ze standardem Hayesa, trzeba podać komendę `ATS0=3`. Jeżeli zamierzasz używać demona *mgetty*, nie jest to konieczne.
2. Skonfigurowanie urządzenia szeregowego za pomocą polecenia *getty*, aby odpowiadało na przychodzące połączenia. Powszechnie stosowaną odmianą *getty* jest *mgetty*.

3. Rozważenie uwierzytelniania. Czy teklienty będą uwierzytelniać się za pomocą PAP, CHAP, czy logowania systemowego?
4. Skonfigurowanie *pppd* jako serwera zgodnie z tym, co napisaliśmy w tym podrozdziale.
5. Rozważenie rutingu. Czy będziesz musiał udostępnić klientom trasę do sieci? Można to zrobić za pomocą skryptu *ip-up*.

Dzwonienie na żądanie

Jeżeli istnieje ruch IP, który ma być przesyłany przez łącze, zestawianie połączenia ze zdalnym hostem można zrealizować przez *dzwonienie na żądanie* (ang. *demand dialing*). Jest ono najbardziej użyteczne, gdy nie możesz na stałe pozostawić swojej linii telefonicznej w stanie połączenia z dostawcą Internetu. Na przykład gdybyś musiał płacić za rozmowy lokalne według czasu, taniej byłoby używać linii tylko wtedy, gdy jest potrzebna, i rozłączać się, gdy nie korzystasz z Internetu.

Tradycyjne rozwiązania w Linuksie wykorzystywały polecenie *diald*, które działało dobrze, ale miało nieco skomplikowaną konfigurację. Wersje 2.3.0 i nowsze demona PPP mają wbudowaną obsługę dzwonienia na żądanie i konfiguruje się ją dość łatwo. Aby działała, musisz użyć także nowszego jądra. Nadają się wszystkie jądra nowsze niż 2.0.

Aby skonfigurować demonowi *pppd* dzwonienie na żądanie, wystarczy dodać opcje w swoim pliku *options* lub w wierszu poleceń *pppd*. Poniższa tabela stanowi skrótowy opis opcji związanych z dzwonieniem na żądanie:

Opcja	*Opis*
`demand`	Ta opcja mówi, że łącze PPP powinno być przełączone do trybu dzwonienia na żądanie. Zostanie utworzone urządzenie sieciowe PPP, ale polecenie `connect` nie będzie używane, dopóki z lokalnego hosta nie zostanie wysłany datagram. Ta opcja jest obowiązkowa, aby dzwonienie na żądanie działało.
`archive-filter` *`wyra□enie`*	Ta opcja pozwala ci określić, które pakiety danych są uznawane za aktywny ruch. Wszelki ruch pasujący do zadanej reguły będzie restartował zegar dzwonienia na żądanie, a *pppd* będzie dalej czekać, zanim rozłączy linię. Składnia filtru została zapożyczona z polecenia *tcpdump*. Domyślny filtr pasuje do wszystkich datagramów.
`holdoff` *`n`*	Ta opcja pozwala na określenie minimalnego czasu (w sekundach), jaki należy odczekać przed rozłączeniem linii, jeżeli nie ma transmisji. Jeżeli połączenie nie działa, a *pppd* myśli, że jest ono cały czas aktywne, zostanie ono ponownie uruchomione, kiedy upłynie czas określony tą opcją. Nie dotyczy ona jednak ponownego połączenia po upłynięciu czasu jałowego oczekiwania.
`idle` *`n`*	Jeżeli ta opcja jest skonfigurowana, *pppd* rozłączy linię, gdy upłynie podany czas. Czasy jałowego oczekiwania są określane w sekundach. Każdy nowy pakiet danych zeruje ten licznik.

Prosta konfiguracja dzwonienia na żądanie mogłaby wyglądać jakoś tak:

```
demand
holdoff 60
idle 180
```

Ta konfiguracja powoduje włączenie dzwonienia na żądanie, odczekanie 60 sekund przed ponownym zestawieniem nieudanego połączenia i rozłączenie, jeżeli w ciągu 180 sekund nie pojawią się żadne aktywne dane na łączu.

Stałe połączenie telefoniczne

Połączenie stałe oznacza, że linia jest cały czas w stanie aktywności, czy li jest podłączona do sieci. Istnieje niewielka różnica pomiędzy dzwonieniem na żądanie a połączeniem stałym. Połączenie stałe jest automatycznie zestawiane zaraz po uruchomieniu demona PPP, a gdy linia telefoniczna obsługująca łącze ulegnie rozłączeniu, podejmowana jest ponowna próba połączenia. Połączenie stałe gwarantuje, że łącze jest dostępne przez cały czas dzięki automatycznemu odtwarzaniu uszkodzonego połączenia.

Możesz być w tej szczęśliwej sytuacji, że nie musisz płacić za rozmowy telefoniczne. Być może są to rozmowy lokalne i dla tego są darmowe, albo są opłacone przez twoją firmę. Opcja stałego połączenia jest niezmiernie przydatna w takiej sytuacji. Jeżeli płacisz za swoje rozmowy, musisz być bardziej ostrożny. Jeżeli płacisz za czas na linii, stałe z całą pewnością nie jest dla ciebie, chyba że naprawdę musisz je mieć przez dwadzieścia cztery godziny na dobę. Jeżeli płacisz za nawiązanie połączenia, a nie za czas trwania rozmowy, musisz zabezpieczyć się przed sytuacjami, które mogą powodować, że twój modem będzie bez końca nawiązywał nowe połączenia. Demon *pppd* ma taką opcję.

Aby uruchomić stałe połączenie telefoniczne, musisz dołączyć opcję `persist` w jednym z plików opcji *pppd*. To wystarczy, aby *pppd* automatycznie wywoływało polecenie określone w opcji `connect`, które odtworzy połączenie w razie jego zerwania. Jeżeli martwisz się o zbyt częste dzwonienie modemu (w przypadku, gdy modem lub serwer po drugiej stronie zostaną uszkodzone), możesz użyć opcji `holdoff`, aby ustawić minimalny czas, jaki *pppd* musi odczekać przed próbą ponownego połączenia. Opcja ta nie rozwiązuje problemu utraty pieniędzy za połączenia w czasie awarii, ale przynajmniej redukuje sumę.

Typowa konfiguracja mogłaby mieć ustawione następujące opcje związane ze stałym połączeniem:

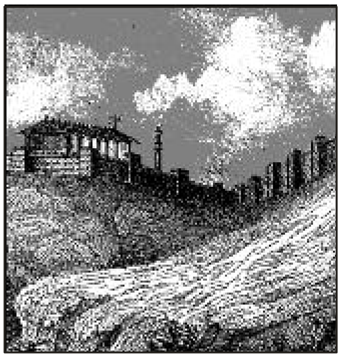
```
persist
holdoff 600
```

Czas oczekiwania na ponowne wybranie numeru jest określony w sekundach. W naszym przykładzie *pppd* czeka pełne pięć minut od momentu zerwania połączenia, zanim zacznie ponownie dzwonić.

Stałe połączenie telefoniczne może współwystąpić z dzwonieniem na żądanie. Aby tak się stało, należy użyć opcji `idle` do rozłączania linii, gdy jest nieaktywna przez zadany okres czasu. Nie wydaje nam się, by wiele osób chciało stosować to rozwiązanie, ale scenariusz ten jest w skrócie opisany na stronie podręcznika elektronicznego *pppd*, gdybyś chciał go wykorzystać.

9

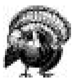
Firewall TCP/IP



Bezpieczeństwo jest coraz ważniejsze zarówno dla firm, jak i osób prywatnych. Internet daje wszystkim doskonałe narzędzia do rozpowszechniania informacji o sobie i uzyskiwania informacji od innych, ale równocześnie niesie ze sobą zagrożenia, których wcześniej nie było: przestępstwa komputerowe, kradzież informacji i złośliwe zniszczenia.

Nieuprawnione i nieuczciwe osoby, które uzyskują dostęp do systemu komputerowego, mogą zgadnąć hasła czy wykorzystać błędy i naturalne zachowanie pewnych programów, aby założyć sobie konto na danym komputerze. Gdy już mogą się zalogować, mają dostęp do różnych informacji, nawet do ważnych informacji handlowych, takich jak plany marketingowe, szczegóły dotyczące nowego projektu czy bazy danych o klientach, które mogą wykorzystać na szkodę ich właściciela. Uszkodzenie lub modyfikacja tego typu danych może narazić na poważne kłopoty firmę.

Najbezpieczniejszym sposobem uniknięcia takich powszechnych zagrożeń jest uniemożliwienie dostępu do sieci osobom nieuprawnionym. Z pomocą przychodzą tutaj firewalle.



Stworzenie bezpiecznych firewalli jest sztuką. Wymaga dobrego zrozumienia technologii, ale co równie ważne, wymaga zrozumienia filozofii sięgającej poza ich konstrukcję. Nie będziemy tu opisywali wszystkiego, co musisz wiedzieć. Radzimy, byś wykonał pewne dodatkowe badania, zanim zaufasz jakiejś szczególnej architekturze firewalla, również tej, którą tutaj pokazujemy.

Istnieje wystarczająco dużo materiału na temat konfiguracji i budowy firewalla, by wypełnić nim całą książkę; do naprawdę doskonałych źródeł należą między innymi:

Building Internet Firewalls

autorstwa D. Chapmana i E. Zwicky (O'Reilly; wyd. polskie nakładem Wydawnictwa RM – w przygotowaniu). Podręcznik wyjaśniający, jak utworzyć i zainstalować firewalle w Uniksie, Linuksie i Windows NT oraz jak skonfigurować usługi internetowe do pracy z firewallami.

Firewalls and Internet Security
: autorstwa W. Cheswicka i S. Bellovina (wyd. Addison Wesley). Ta książka omawia filozofię budowy i implementacji firewalli.

W tym rozdziale skupimy się na zagadnieniach technicznych, specyficznych dla Linuksa. Później pokażemy przykładową konfigurację firewalla, która powinna służyć jako użyteczny punkt wyjścia do własnej konfiguracji, ale jak to bywa w zagadnieniach bezpieczeństwa, nigdy nikomu nie ufaj. Dwa razy sprawdź projekt, upewnij się, że go rozumiesz, a następnie zmodyfikuj tak, aby pasował do twoich potrzeb. Pewność to bezpieczeństwo.

Metody ataku

Dla administratora sieci ważne jest, by rozumiał istotę potencjalnych ataków zagrażających bezpieczeństwu komputera. Pokrótce opiszemy najważniejsze typy ataków, tak byś lepiej zrozumiał, przed czym chroni cię firewall IP w Linuksie. Aby być pewnym, że jesteś w stanie zabezpieczyć swoją sieć przed innymi typami ataków, powinieneś sięgnąć po dodatkową lekturę. Oto najważniejsze metody ataku i sposoby zabezpieczania się przed nimi:

Nieautoryzowany dostęp
: Oznacza po prostu, że ludzie, którzy nie powinni korzystać z usług oferowanych przez twój komputer, są w stanie się do niego podłączyć i z nich korzystać. Na przykład ludzie spoza firmy mogą próbować połączyć się z komputerem obsługującym księgowość twojej firmy lub z twoim serwerem NFS.

: Istnieją różne sposoby uniknięcia tego ataku. Trzeba precyzyjnie określić, kto może mieć dostęp do danych usług. Możesz zabronić dostępu do sieci wszystkim poza wyznaczonymi przez ciebie osobami.

Wykorzystanie znanych dziur w programach
: Wczasach kiedy powstawały niektóre programy i usługi sieciowe, nie uwzględniano jeszcze rygorystycznych zasad bezpieczeństwa. Te właśnie są z natury bardziej podatne na zagrożenia. Usługi zdalne BSD (rlogin, rexec itp.) są tu doskonałym przykładem.

: Najlepszym sposobem na zabezpieczenie się przed tego typu atakiem jest wyłączenie wszelkich podatnych usług lub znalezienie alternatywy. W przypadku Open Source czasem możliwe jest załatanie dziury w programie.

Odmowa obsługi
: Ataki typu odmowa obsługi powodują, że usługa lub program przestają działać lub nie pozwalają innym z siebie korzystać. Może to być spowodowane wysyłaniem w warstwie sieciowej starannie przygotowanych, złośliwych datagramów, które powodują awarie połączeń sieciowych. Ataki mogą być też realizowane w warstwie aplikacji, gdzie starannie przygotowane polecenia aplikacji podane programowi powodują, że staje się on nadzwyczaj zajęty lub przestaje działać.

Uniemożliwienie podejrzanym pakietom sieciowym dotarcia do twojego hosta oraz zapobiegnięcie uruchamianiu podejrzanych poleceń i żądań są najlepszymi sposobami na zminimalizowanie ryzyka ataku od mowy obsługi. Warto dobrze znać metody ataku, a więc powinieneś sam dokształcać się na temat wszystkich nowych ataków, gdy ich opis zostanie opublikowany.

Podszywanie się

Ten typ ataku powoduje, że host lub aplikacja naśladują działanie innego. Zwykle atakujący udaje niewinny host, przesyłając sfałszowane adresy IP w pakietach sieciowych. Na przykład dobrze udokumentowany sposób wykorzystania usługi rlogin BSD stosuje tę metodę do udawania połączenia TCP z innego hosta. Robi to, odgadując numery kolejnych pakietów TCP.

Aby zabezpieczyć się przed tego typu atakiem, weryfikuj wiarygodność datagramów i poleceń. Wyłącz możliwość rutowania datagramów o złym adresie źródłowym. Wprowadź nieprzewidywalność do mechanizmu kontroli połączenia, na przykład stosowanie kolejnych numerów TCP lub alokację dynamicznych adresów portów.

Podsłuchiwanie

Jest to najprostszy typ ataku. Host jest skonfigurowany na „słuchanie" i zbieranie danych nie należących do niego. Dobrze napisane programy podsłuchujące mogą odczytać z połączeń sieciowych nazwy użytkowników i hasła. Sieci rozgłoszeniowe, takie jak Ethernet, są szczególnie podatne na tego typu atak.

Lepiej więc unikaj rozwiązań opartych o sieci rozgłoszeniowe i wprowadzaj szyfrowanie danych.

Firewalle IP są bardzo użyteczne; są w stanie zapobiec nieautoryzowanym dostępom, odmowom obsługi w warstwie sieciowej i atakom przez podszywanie się lub znacznie zmniejszyć ryzyko ich wystąpienia. Nie zbyt dobrze zabezpieczają przed wykorzystywaniem dziur w usługach sieciowych czy programach oraz nie zapobiegają podsłuchiwaniu.

Co to jest firewall

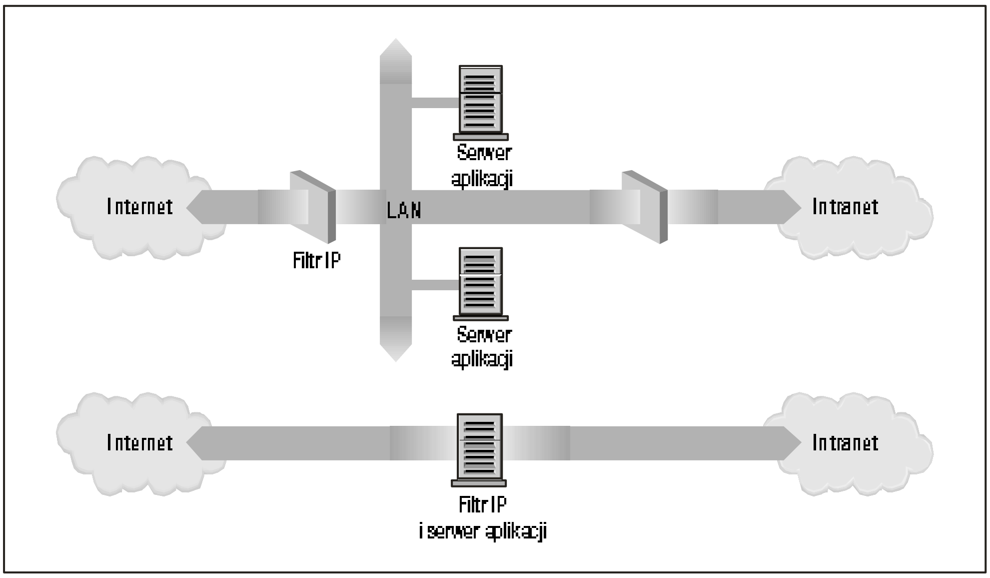
Firewall to bezpieczny i zaufany komputer, który jest umieszczony pomiędzy siecią prywatną a siecią publiczną*. Komputer-firewall jest skonfigurowany w oparciu o zestaw reguł, które określają, jaki ruch sieciowy może być przepuszczony, a jaki ma być blokowany lub odrzucany. W niektórych dużych firmach możesz znaleźć nawet firewalle umieszczone wewnątrz sieci firmowej, które oddzielają ważne obszary firmy od innych pracowników. Wiele przestępstw komputerowych zdarza się wewnątrz firmy, a nie jest powodowane atakami z zewnątrz.

Firewalle można budować na różne sposoby. Najbardziej wyrafinowane konstrukcje wykorzystują kilka oddzielnych komputerów i są znane pod nazwą *sieciowy dzie-*

* Określenie *firewall* (ang.) zostało przejęte od urządzenia używanego do zabezpieczania ludzi przed ogniem. Firewall to osłona z ognioodpornego materiału umieszczana pomiędzy miejscem zagrożonym pożarem a człowiekiem, którego ma chronić.

lonej (ang. *perimeter network*). Dwie maszyny działają jako „filtry" pozwalające tylko na przepuszczanie pewnych typów ruchu sieciowego. Pomiędzy tymi maszynami znajdują się serwery sieciowe, takie jak gateway poczty czy serwer proxy WWW. Konfiguracja taka może być bardzo bezpieczna i łatwo pozwala na osiągnięcie znacznej kontroli nad tym, kto łączy się zarówno z zewnątrz do wewnątrz, jak i z wewnątrz na zewnątrz. Tego typu konfiguracja może być stosowana w dużych firmach.

Zwykle jednak firewalle to pojedyncze maszyny pełniące wszystkie te funkcje. Są one nieco mniej bezpieczne, ponieważ jeżeli w maszynie pełniącej rolę firewalla znajdzie się dziura, która pozwala na uzyskanie dostępu do niej, to bezpieczeństwo całej sieci zostaje naruszone. Jednak tego typu firewalle są tańsze i prostsze w zarządzaniu niż opisane wcześniej, bardziej wyrafinowane rozwiązania. Rysunek 9-1 pokazuje dwie najpopularniejsze konfiguracje firewalla.



Rysunek 9-1. Dwie podstawowe architektury firewalli

Jądro Linuksa udostępnia szereg wbudowanych funkcji pozwalających mu na pracę w roli firewalla IP. Implementacja sieci zawiera kod do wykonywania filtrowania IP na szereg różnych sposobów i udostępnia mechanizm pozwalający na dokładne skonfigurowanie zasad, jakimi chcesz się kierować przy takim filtrowaniu. Firewall w Linuksie jest wystarczająco elastyczny, by można go było zastosować w obu konfiguracjach pokazanych na rysunku 9-1. Oprogramowanie firewalla w Linuksie udostępnia dwie inne przydatne funkcje, które omówimy w oddzielnych rozdziałach: liczenie ruchu IP (rozdział 10, *Liczenie ruchu IP*) i maskowanie IP (rozdział 11, *Maskowanie IP i translacja adresów sieciowych*).

Co to jest filtrowanie IP

Filtrowanie IP to prosty mechanizm decydujący o tym, które typy datagramów IP mają być przetwarzane normalnie, a które mają być odrzucone. Przez *odrzucenie* (ang. *discard*) rozumiemy, że datagramy są usuwane i zupełnie ignorowane, tak jakby nigdy nie zostały odebrane. Możesz wskazać wiele różnych kryteriów określających, które datagramy chcesz filtrować. Oto kilka przykładów:

- typ protokołu: TCP, UDP, ICMP itp.;
- numer gniazda (dla TCP/UDP);
- typ datagramu: SYN/ACK, dane, ICMP Echo Request itp.;
- adres źródłowy datagramu: skąd pochodzi;
- adres docelowy datagramu: dokąd jest wysyłany;

Ważne jest zrozumienie w tym miejscu, że filtrowanie IP jest funkcją warstwy sieciowej. Oznacza to, że nie ma ono nic wspólnego z aplikacją wykorzystującą połączenia sieciowe, a dotyczy tylko samych połączeń. Na przykład możesz zabronić użytkownikom dostępu do swojej sieci wewnętrznej przez standardowy port telnet, ale jeżeli opierasz się na samym filtrowaniu IP, nie możesz spowodować, żeby przestali używać programu telnet na porcie, który przepuszczasz przez swój firewall. Aby uniknąć tego typu problemów, zastosuj serwery proxy dla każdej usługi, którą chcesz przepuścić przez firewall. Serwery proxy rozumieją aplikacje, dla których mają pełnić rolę pośredników i w ten sposób mogą zapobiec nadużyciom, polegającym na przykład na wykorzystywaniu telnetu do połączenia z portem WWW poprzez firewall. Jeżeli twój firewall obsługuje WWW proxy, połączenia telnet do tej usługi będą zawsze obsługiwane przez proxy i dopuszczalne będą tylko żądania HTTP. Istnieje szereg programów typu serwer proxy. Niektóre są darmowe, ale jest też wiele komercyjnych. *Firewall-HOWTO* omawia jeden z popularnych zestawów takich programów, których tu nie przedstawiamy, ponieważ wykracza to poza zakres tej książki.

Zestaw reguł filtrowania IP składa się z wielu kombinacji wymienionych powyżej kryteriów. Na przykład wyobraźmy sobie, że chciałbyś pozwolić użytkownikom WWW z sieci browaru wirtualnego na dostęp do Internetu, ale tylko na korzystanie z innych serwerów WWW. Musiałbyś skonfigurować firewall, tak by pozwalał na przekazywanie:

- datagramów z adresem źródłowym z sieci browaru wirtualnego i dowolnym adresem docelowym oraz z portem docelowym 80 (WWW),
- datagramów z adresem docelowym browaru wirtualnego i portem źródłowym 80 (WWW) z dowolnego adresu źródłowego.

Zauważ, że użyliśmy tutaj dwóch reguł. Pozwalamy na wychodzenie naszych danych oraz na przyjmowanie odpowiedzi. W praktyce, jak wkrótce zobaczymy, Linux upraszcza to i pozwala na określenie tych reguł jednym poleceniem.

Skonfigurowanie Linuksa w roli firewalla

Aby stworzyć firewall IP w Linuksie, potrzebne jest jądro z wbudowaną obsługą firewalla IP i odpowiedni program konfiguracyjny. We wszystkich jądrach do serii 2.0 używa się narzędzia *ipfwadm*. Jądra 2.2.x wprowadziły trzecią generację firewalla IP dla Linuksa o nazwie *IP Chains* (łańcuchy IP). Łańcuchy IP używają programu podobnego do *ipfwadm*, ale noszącego go na zwę *ipchains*. Jądra w wersji 2.3.15 i nowsze obsługują czwartą generację firewalli o nazwie *netfilter*. Kod *netfilter* jest wynikiem poważnej zmiany w sposobie obsługi pakietów w Linuksie. *netfilter* jest tworem uniwersalnym, zapewnia bowiem bezpośrednio wsteczną kompatybilność zarówno z *ipfwadm*, jak i *ipchains* oraz nowe alternatywne polecenie *iptables*. W następnych podrozdziałach omówimy różnice pomiędzy tymi trzema rozwiązaniami.

Jądro skonfigurowane z firewallem IP

Jądro Linuksa trzeba odpowiednio skonfigurować, aby obsługiwało firewall IP. Oznacza to po prostu wybór odpowiednich opcji po wywołaniu `make menuconfig` przy kompilacji jądra*. Jak to zrobić, opisaliśmy w rozdziale 3, *Konfigurowanie sprzętu sieciowego*. W jądrach 2.2 powinieneś wybrać następujące opcje:

```
Networking options --->
       [*] Network firewalls
       [*] TCP/IP networking
       [*] IP: firewalling
       [*] IP: firewall packet logging
```

W jądrach 2.4.0 i nowszych powinieneś wybrać poniższe opcje:

```
Networking options --->
   [*] Network packet filtering (replaces ipchains)
       IP: Netfilter Configuration ---?
           .
           <M> Userspace queueing via NETLINK (EXPERIMENTAL)
           <M> IP tables support (required for filtering/masq/NAT)
           <M> limit match support
           <M> MAC address match support
           <M> netfilter MARK match support
           <M> Multiple port match support
           <M> TOS match support
           <M> Connection state match support
           <M> Unclean match support (EXPERIMENTAL)
           <M> Owner match support (EXPERIMENTAL)
           <M> Packet filtering
           <M>   REJECT target support
           <M>   MIRROR target support (EXPERIMENTAL)
           .
           <M> Packet mangling
           <M>   TOS target support
           <M>   MARK target support
           <M> LOG target support
           <M> ipchains (2.2-style) support
           <M> ipfwadm (2.0-style) support
```

* Logowanie pakietów przez firewall jest specjalną funkcją, która zapisuje wiersz informacji o każdym datagramie odpowiadającym regule do specjalnego urządzenia, przez które możesz go zobaczyć.

Narzędzie ipfwadm

Narzędzie *ipfwadm* (*IP Firewall Administration*) jest używane do tworzenia reguł firewalla dla wszystkich jąder starszych od wersji 2.2.0. Składnia polecenia bywa zagmatwana, ponieważ może ono realizować wiele skomplikowanych zadań, ale podamy kilka przykładów popularnych zastosowań.

Narzędzie *ipfwadm* jest zawarte w większości współczesnych dystrybucji Linuksa, ale niekoniecznie standardowo. Mogą istnieć szczególne pakiety oprogramowania, które musisz zainstalować, aby mieć to polecenie. Jeżeli nie ma go w twojej dystrybucji, możesz zdobyć pakiet źródłowy z ośrodka **ftp.xos.nl** z katalogu */pub/linux/ipfwadm/* i skompilować go samodzielnie.

Narzędzie ipchains

Podobnie jak *ipfwadm*, tak *ipchains* może sprawić na początku nieco kłopotów. Udostępnia całą elastyczność *ipfwadm*, ale za pomocą poleceń o znacznie uproszczonej składni, a ponadto oferuje mechanizm „łączenia w łańcuchy" (ang. *chaining*), pozwalający na zarządzanie wieloma zestawami reguł i ich łączenie. Łączenie reguł omówimy w oddzielnym podrozdziale pod koniec tego rozdziału, ponieważ jest to pojęcie zaawansowane.

Polecenie *ipchains* istnieje w większości dystrybucji Linuksa opartych na jądrach 2.2. Gdybyś chciał skompilować je samodzielnie, możesz znaleźć pakiet źródłowy pod adresem: *http://www.rustcorp.com/linux/ipchains/*. W pakiecie tym znajduje się dodatkowy skrypt *ipfwadm-wrapper*, który naśladuje polecenie *ipfwadm*, ale w rzeczywistości wywołuje polecenie *ipchains*. Migracja istniejącej konfiguracji firewalla jest dużo mniej bolesna, jeżeli posiada się taki skrypt.

Narzędzie iptables

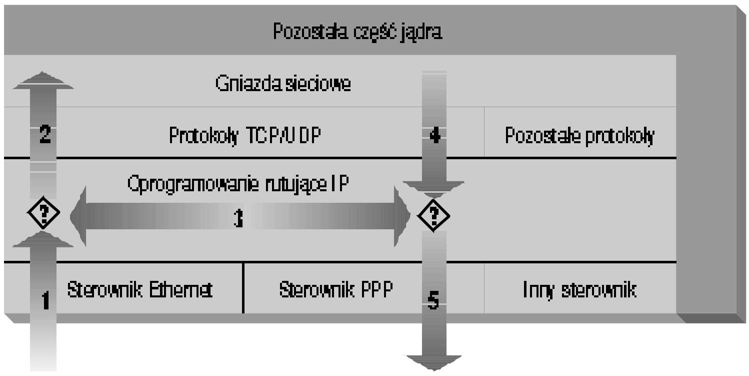
Składnia narzędzia *iptables* jest bardzo podobna do *ipchains*. Różnice wynikają z wprowadzonych udoskonaleń i dają przeprojektowane narzędzie, które jest rozszerzalne przez biblioteki dzielone. Tak jak w przypadku *ipchains*, tak i przy *iptables* podamy skonwertowane przykłady, abyś mógł porównać i zestawić składnię tego i innych poleceń.

Narzędzie *iptables* znajduje się w pakiecie *netfilter*, który jest dostępny pod adresem *http://www.samba.org/netfilter/*. Będzie także zawarte we wszystkich dystrybucjach Linuksa opartych o jądra serii 2.4.

Nieco więcej o *netfilter* powiemy w jednym z następnych podrozdziałów poświęconych tylko temu pakietowi.

Trzy sposoby realizacji filtrowania

Rozważmy, w jaki sposób maszyna unikso wa, czy w praktyce dowolna inna obsługująca ruting IP, przetwarza datagramy IP.



Rysunek 9-2. Etapy przetwarzania datagramu IP

Podstawowe kroki, pokazane na rysunku 9-2, to:

- Datagram IP jest odbierany (1).
- Przychodzący datagram IP jest analizowany w celu ustalenia, czy jest przeznaczony dla tej maszyny.
- Jeżeli datagram jest dla tej maszyny, jest przetwarzany lokalnie (2).
- Jeżeli nie jest dla tej maszyny, w tablicy rutingu jest poszukiwana odpowiednia trasa. Datagram jest przekazywany do odpowiedniego interfejsu lub odrzucany, jeżeli trasy nie można znaleźć (3).
- Datagramy z lokalnych procesów są wysyłane do oprogramowania rutującego w celu przekazania do odpowiedniego interfejsu (4).
- Wychodzący datagram IP jest analizowany w celu ustalenia, czy istnieje dla niego odpowiednia trasa; jeżeli nie, jest odrzucany.
- Datagram IP jest wysyłany (5).

W naszym diagramie, przepływ 1→3→5 przedstawia maszynę rutującą dane pomiędzy hostem w naszej sieci Ethernet a hostem osiągalnym przez łącze PPP. Przepływ 1→2 i 4→5 przedstawia dane przychodzące i wychodzące z programu sieciowego działającego na naszym hoście lokalnym. Przepływ 4→3→2 przedstawia przepływ danych przez połączenie pętli zwrotnej. Oczywiście dane przepływają w obie strony, do i z urządzeń sieciowych. Znak zapytania na diagramie przedstawia punkty, gdzie warstwa IP podejmuje decyzje co do rutingu.

Firewall jądra Linuksa może stosować filtry na różnych etapach tego procesu. To znaczy, że możesz filtrować datagramy IP przychodzące do twojej maszyny, albo te

datagramy, które są przez nią przekazywane, albo też te, które są gotowe do wysłania.

W programach *ipfwadm* i *ipchains* reguła wejściowa (Input) dotyczy przepływu 1 na diagramie, reguła przekazywania (Forwarding) – przepływu 3, a reguła wyjściowa (Output) – przepływu 5. Zobaczymy później, przy omawianiu *netfilter*, że punkty przechwytywania zmieniły się tak, że reguła wejściowa dotyczy przepływu 2, a reguła wyjściowa – przepływu 4. Ma to istotny wpływ na tworzenie reguł, ale ogólne zasady pozostają takie same dla wszystkich wersji firewalla w Linuksie.

Na pierwszy rzut oka może to wyglądać na niepotrzebną komplikację, ale zapewnia elastyczność, która pozwala na tworzenie bardzo wyrafinowanych i wydajnych konfiguracji.

Oryginalny firewall IP (jądra 2.0)

Pierwsza generacja obsługi firewalla IP w Linuksie pojawiła się w serii jąder 1.1. Było to przeniesienie ipfw z BSD dokonane przez Alana Coxa. Obsługa firewalla w jądrach serii 2.0, określana mianem drugiej generacji, została rozszerzona przez Josa Vos, Pauline Middelink i innych.

Korzystanie z ipfwadm

Polecenie *ipfwadm* było narzędziem konfiguracyjnym dla drugiej generacji firewalla IP w Linuksie. Najlepiej jest opisać użycie *ipfwadm* na przykładach. Na początek zakodujmy pokazany wcześniej przykład.

Prosty przykład

Załóżmy, że mamy w naszej firmie sieć i używamy firewalla na komputerze z Linuksem, przez który łączymy naszą sieć z Internetem. Przyjmijmy też, że chcemy, aby użytkownicy sieci mieli dostęp do serwerów WWW w Internecie, ale nie dopuszczamy żadnego innego ruchu.

Zdefiniujemy regułę przekazywania pozwalającą na przepuszczanie na zewnątrz datagramów o adresie źródłowym należącym do naszej sieci i gnieździe docelowym 80 oraz na przekazywanie przez firewall odpowiedzi przesyłanych z powrotem.

Załóżmy, że nasza sieć ma 24-bitową maskę (klasa C) i adres 172.16.1.0. Reguły wyglądają tak:

```
# ipfwadm -F -f
# ipfwadm -F -p deny
# ipfwadm -F -a accept -P tcp -S 172.16.1.0/24 -D 0/0 80
# ipfwadm -F -a accept -P tcp -S 0/0 80 -D 172.16.1.0/24
```

Argument wiersza poleceń *–F* mówi *ipfwadm*, że jest to reguła przekazywania. Pierwsze polecenie mówi *ipfwadm*, aby usunął wszystkie dotychczasowe reguły przekazywania ze swojej konfiguracji. W ten sposób rozpoczynamy od znanego stanu.

Druga reguła określa domyślną politykę przekazywania. Mówimy jądru, by odrzucało lub nie pozwalało na przekazywanie datagramów IP. Bardzo ważne jest ustawienie polityki domyślnej, ponieważ opisuje ona, co się stanie z datagramami, które nie są w żaden szczególny sposób obsługiwane przez inne reguły. Zwykle konfigurując firewall, będziesz ustawiał domyślną politykę na „odmowę”, tak jak to pokazano tutaj, po to, aby przez firewalla przechodził tylko dopuszczalny ruch.

Trzecia i czwarta reguła implementują nasze wymaganie: – trzecie polecenie pozwala na wysyłanie naszych datagramów, a czwarte – na przyjmowanie odpowiedzi.

Przyjrzyjmy się kolejno argumentom:

-F

Jest to reguła przekazująca.

-a accept

Reguła z dopisaną polityką „akceptowania” oznacza, że będziemy przekazywać wszystkie datagramy, które do niej pasują.

-P tcp

Ta reguła dotyczy datagramów tcp (w przeciwieństwie do UDP lub ICMP).

-S 172.16.1.0/24

Adres źródłowy musi mieć pierwsze 24 bity odpowiadające adresowi sieci 172.16.1.0.

-D 0/0 80

Adres docelowy musi mieć zero bitów pasujących do adresu 0.0.0.0. Tak na prawdę jest to skrótowy zapis „wszystkiego”. Port docelowy to 80, co w tym przypadku oznacza WWW. Do opisania portu możesz użyć także wszelkich wpisów znajdujących się w pliku */etc/services*, a więc `-D 0/0 www` działałoby równie dobrze.

ipfwadm wymaga maski sieci w postaci, która może nie być ci znana. Zapis `/nn` oznacza liczbę istotnych bitów w podanym adresie lub rozmiar maski. Bity są zawsze liczone od lewej do prawej. W tabeli 9-1 podano często spotykane przykłady masek.

Tabela 9-1. Często spotykane maski sieci

Maska sieci	*Bity*
255.0.0.0	8
255.255.0.0	16
255.255.255.0	24
255.255.255.128	25
255.255.255.192	26
255.255.255.224	27
255.255.255.240	28
255.255.255.248	29
255.255.255.252	30

Wcześniej wspomnieliśmy, że *ipfwadm* implementuje małą sztuczkę, która ułatwia dodawanie tego typu reguł. Ta sztuczka to opcja *–b*, która sprawia, że polecenie jest dwukierunkowe.

Opcja dwukierunkowości pozwala na połączenie naszych dwóch reguł w jedną:

```
# ipfwadm -F -a accept -P tcp -S 172.16.1.0/24 -D 0/0 80 -b
```

Ważna poprawka

Przyjrzyj się bliżej naszemu zestawowi reguł. Czy widzisz, że wciąż nie ma zabezpieczenia przed jedną metodą ataku, którą ktoś z zewnątrz może wykorzystać do pokonania naszego firewalla?

Nasze reguły pozwalają na przyjmowanie z zewnątrz wszystkich datagramów z portem źródłowym 80. Uwaga! Będą do nich należały także datagramy z ustawionym bitem SYN! Bit SYN oznacza, że jest to datagram TCP z żądaniem połączenia. Jeżeli osoba z zewnątrz miałaby uprzywilejowany dostęp do swojego hosta, mogłaby połączyć się przez nasz firewall z dowolnym z naszych hostów, pod warunkiem, że używają one portu 80. Nie to chcieliśmy osiągnąć.

Na szczęście istnieje rozwiązanie tego problemu. Polecenie *ipfwadm* posiada inną opcję, która pozwala nam budować reguły odfiltrowujące datagramy z ustawionym bitem SYN. Zmieńmy nasz przykład tak, aby uwzględniał tę regułę:

```
# ipfwadm -F -a deny -P tcp -S 0/0 80 -D 172.16.10.0/24 -y
# ipfwadm -F -a accept -P tcp -S 172.16.1.0/24 -D 0/0 80 -b
```

Opcja *–y* powoduje, że reguła pasuje tylko wtedy, jeżeli bit SYN w datagramie jest ustawiony. A więc nasza nowa reguła mówi: „Nie przepuszczaj żadnych datagramów TCP przeznaczonych dla naszej sieci, które pochodzą z jakiegoś miejsca i mają port źródłowy 80 i ustawiony bit SYN” albo „Nie przepuszczaj żadnych żądań od hostów na port 80”.

Dlaczego umieściliśmy tę szczególną regułę *przed* regułą główną? Reguły firewalla działają tak, że są dopasowywane kolejno. Obie reguły będą pasowały do datagramów, których nie chcemy przepuścić, a więc musimy być pewni, że reguła `deny` jest przed regułą `accept`.

Listowanie naszych reguł

Po wprowadzeniu naszych reguł możemy je wylistować, wywołując *ipfwadm* w następujący sposób:

```
# ipfwadm -F -l
```

To polecenie da w wyniku wszystkie skonfigurowane reguły przekazywania. Rezultat powinien być podobny do tego:

```
# ipfwadm -F -l
IP firewall forward rules, default policy: accept
type   prot   source          destination         ports
deny   tcp    anywhere        172.16.10.0/24      www -> any
acc    tcp    172.16.1.0/24   anywhere            any -> www
```

Polecenie *ipfwadm* będzie próbowało tłumaczyć numer portu na nazwę usługi za pomocą pliku */etc/services*, o ile istnieje w nim wpis.

W domyślnie pokazywanym wyniku brakuje kilku ważnych dla nas szczegółów. Nie widać tam mianowicie działania argumentu `-y`. Polecenie *ipfwadm* potrafi pokazać dokładniejszy wynik, jeżeli podamy także opcję `-e` (wynik rozszerzony). Nie pokażemy całego wyniku, ponieważ jest zbyt szeroki i nie mieści się na stronie, ale zawiera kolumnę `opt` (opcje), która pokazuje opcję *-y* kontrolującą pakiety SYN:

```
# ipfwadm -F -l -e
IP firewall forward rules, default policy: accept
pkts bytes type prot opt  tosa tosx ifname ifaddress  source         ...
   0     0 deny tcp  --y- 0xFF 0x00 any    any        anywhere       ...
   0     0 acc  tcp  b--- 0xFF 0x00 any    any        172.16.1.0/24 ...
```

Bardziej skomplikowany przykład

Poprzedni przykład był prosty. Nie wszystkie usługi sieciowe są tak łatwe do skonfigurowania jak WWW. W rzeczywistości typowa konfiguracja firewalla będzie dużo bardziej złożona. Przyjrzyjmy się innemu powszechnie spotykanemu przykładowi, tym razem FTP. Chcemy, aby użytkownicy naszej wewnętrznej sieci mogli logować się do serwerów FTP w Internecie po to, by odczytywać i zapisywać pliki. Nie chcemy jednak, aby ludzie z Internetu logowali się do naszych serwerów FTP.

Wiemy, że FTP używa dwóch portów TCP: portu 20 (ftp-data) i portu 21 (ftp), a więc:

```
# ipfwadm -a deny -P tcp -S 0/0 20 -D 172.16.1.0/24 -y
# ipfwadm -a accept -P tcp -S 172.16.1.0/24 -D 0/0 20 -b
#
# ipfwadm -a deny -P tcp -S 0/0 21 -D 172.16.1.0/24 -y
# ipfwadm -a accept -P tcp -S 172.16.1.0/24 -D 0/0 21 -b
```

Dobrze? Nie całkiem. Serwery FTP mogą działać w dwóch różnych trybach: w trybie biernym (ang. *passive mode*) i czynnym (ang. *active mode*)*. W trybie biernym serwer FTP oczekuje na połączenie od klienta. W trybie czynnym serwer realizuje połączenie do klienta. Tryb czynny jest zwykle domyślny. Różnice ilustruje rysunek 9-3.

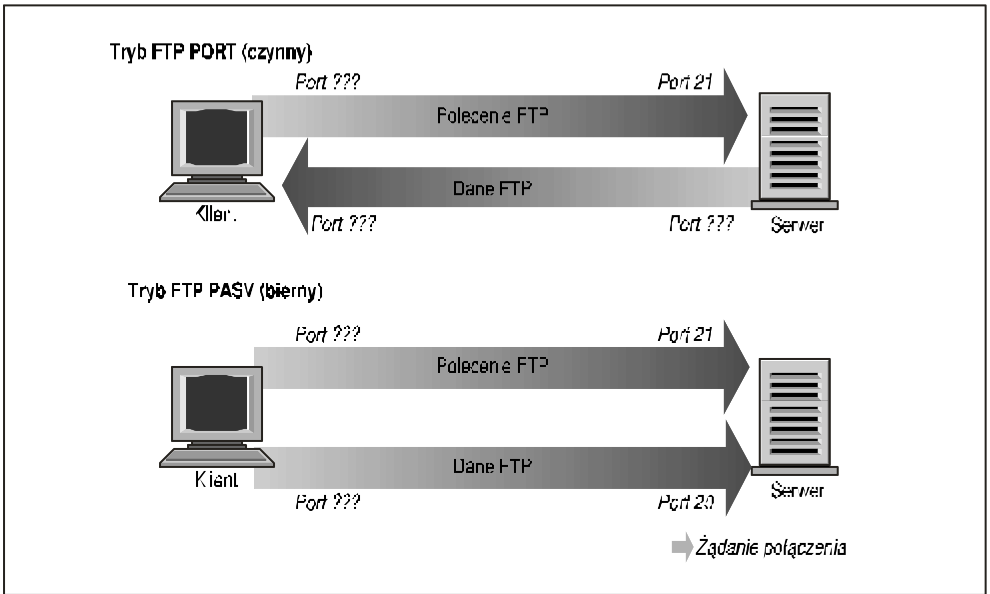
Wiele serwerów FTP działających w trybie czynnym tworzy połączenie z portu 20, co nieco upraszcza sprawę, ale niestety nie wszystkie tak robią**.

Jakie to ma jednak dla nas znaczenie? Przyjrzyjmy się naszej regule dla portu 20, czyli – portu FTP-data. Obecna reguła zakłada, że połączenie będzie inicjowane przez naszego klienta do serwera. Będzie to działało, jeżeli użyjemy trybu biernego. Ale bardzo trudno jest nam skonfigurować poprawną regułę pozwalającą na użycie trybu czynnego, ponieważ nie jesteśmy w stanie z góry przewidzieć, jakie porty będą używane. Jeżeli otworzymy firewall, pozwalając na połączenia przychodzące na dowolny port, narazimy naszą sieć na atak poprzez wszystkie usługi przyjmujące połączenia.

* Tryb czynny FTP jest czymś nieintuicyjnie włączanym za pomocą polecenia *PORT*. Tryb bierny jest włączany za pomocą polecenia *PASV*.

** Demon *ProFTPd* jest dobrym przykładem serwera FTP, który działa inaczej, przynajmniej w starszych wersjach.

W tej sytuacji najlepiej jest wymusić na naszych użytkownikach pracę w trybie biernym. Większość serwerów FTP i wiele klientów FTP działa w ten sposób. Popularny klient *ncftp* także obsługuje tryb bierny, ale może wymagać niewielkiej zmiany w konfiguracji, by był to jego tryb domyślny. Wiele przeglądarek WWW, takich jak Netsccape, także obsługuje bierny tryb FTP, a więc znalezienie odpowiedniego oprogramowania nie powinno być zbyt trudne. Można też postąpić zupełnie inaczej: użyć serwera proxy FTP, który będzie przyjmował połączenia z sieci wewnętrznej i realizował połączenia z siecią zewnętrzną.



Rysunek 9-3. Tryby serwera FTP

Przy projektowaniu firewalla prawdopodobnie napotkasz niejeden taki problem. Powinieneś zawsze dokładnie przeanalizować, jak naprawdę działa dana usługa, by być pewnym, że umieściłeś odpowiedni zestaw reguł w odpowiednim miejscu. Konfiguracja prawdziwego firewalla może być dość skomplikowana.

Podsumowanie argumentów ipfwadm

Polecenie *ipfwadm* ma wiele różnych argumentów odnoszących się do konfiguracji firewalla IP. Ogólna składnia jest następująca:

```
ipfwadm kategoria polecenie parametry [opcje]
```

Przyjrzyjmy się kolejno każdemu z członów.

Kategorie

Musi być podana jedna i tylko jedna z poniższych kategorii. Kategoria mówi firewallowi, jakiego typu regułę konfigurujesz:

-I

Reguła wejściowa.

-O

Reguła wyjściowa.

-F

Reguła przekazywania.

Polecenia

Przynajmniej jedno z poniższych poleceń musi być podane i musi się ono odnosić do określonej wcześniej kategorii. Polecenia mówią firewallowi, co ma robić.

-a [polityka]

Dodanie nowej reguły.

-i [polityka]

Wstawienie nowej reguły.

-d [polityka]

Usunięcie istniejącej reguły.

-p polityka

Ustawienie polityki domyślnej.

-l

Wylistowanie wszystkich istniejących reguł.

-f

Usunięcie wszystkich istniejących reguł.

Polityki istotne dla firewalla IP i ich znaczenie jest następujące:

accept

Pozwala na odbiór, przekazywanie lub wysyłanie pasujących datagramów.

deny

Nie pozwala na odbiór, przekazywanie lub wysyłanie pasujących datagramów.

reject

Nie pozwala na odbiór, przekazywanie lub wysyłanie pasujących datagramów i wysyła komunikat błędu ICMP do hosta, który przesłał datagram.

Parametry

Musi być podany przynajmniej jeden z poniższych parametrów. Używaj parametrów do określania datagramów, których dotyczą reguły:

-P protokół

Może mieć wartość TCP, UDP, ICMP lub all. Przykład:

```
-P tcp
```

-S adres/maska/[port]

· ródłowy adres IP, do którego pasuje ta reguła. Jeżeli nie podasz maski sieci, zostanie przyjęta maska „/32". Opcjonalnie możesz określić, którego portu dotyczy

reguła. Musisz także podać protokół za pomocą opisanego wyżej argumentu *–P*, aby ta opcja zadziałała. Jeżeli nie podasz portu lub zakresu portów, przyjmuje się, że wszystkie porty pasują. Porty mogą być podane w postaci nazwy zgodnej z wpisem w */etc/services*. W przypadku protokołu ICMP pole portu jest używane do oznaczenia typu datagramu ICMP. Możliwe jest podanie zakresu portów, a służy do tego następująca składnia: `pierwszyport:ostatniport`. Oto przykład:

```
-S 172.29.16.1/24 ftp:ftp-data
```

-D adres/maska/[port]
: Określenie adresu docelowego IP, do którego pasuje ta reguła. Adres docelowy jest zapisywany na tej samej zasadzie co adres źródłowy opisany poprzednio. Oto przykład:

```
-D 172.29.16.1/24 smtp
```

-V adres
: Określenie adresu interfejsu sieciowego, na którym pakiet jest odbierany (*-I*) lub z którego jest wysyłany (*-O*). Pozwala to na stworzenie reguł dotyczących tylko niektórych interfejsów sieciowych komputera. Oto przykład:

```
-V 172.29.16.1
```

-W nazwa
: Określenie nazwy interfejsu sieciowego. Ten argument działa w ten sam sposób co *–V*, ale podajesz nazwę urządzenia zamiast adresu. Oto przykład:

```
-W ppp0
```

Argumenty opcjonalne

Te argumenty są czasem bardzo przydatne:

-b
: Jest używany dla trybu dwukierunkowego. Do tej opcji pasuje ruch w obie strony pomiędzy zadanymi adresami źródłowym i docelowym. Zaoszczędza ci ona tworzenia dwóch reguł: jednej do wysyłania i drugiej do odbierania.

-o
: Pozwala na zapisywanie pasujących datagramów do logu jądra. Wszelkie datagramy pasujące do reguły będą zapisywane jako komunikaty jądra. Jest to użyteczna opcja do wykrywania nieautoryzowanego dostępu.

-y
: Ta opcja jest używana do filtrowania połączeniowych datagramów TCP. Dzięki niej reguła filtruje tylko datagramy podejmujące próbę zestawienia połączeń TCP. Pasować będą jedynie datagramy posiadające ustawiony bit SYN i wyzerowany bit ACK. Jest to użyteczna opcja do filtrowania prób połączeń TCP i ignorowania innych protokołów.

-k
: Jest używana do filtrowania datagramów-potwierdzeń TCP (ang. *acknowledgement*). Ta opcja powoduje, że do reguły pasują tylko datagramy będące po-

twierdzeniem odbioru pakietów próbujących zestawić połączenie TCP. Będą pasować jedynie datagramy, które mają ustawiony bit ACK. Opcja ta jest użyteczna do filtrowania prób połączeń TCP i ignorowania wszystkich pozostałych protokołów.

Typy datagramów ICMP

Każde polecenie konfiguracyjne firewalla pozwala ci określać typy datagramów ICMP. W odróżnieniu od portów TCP i UDP, nie ma odpowiedniego pliku konfiguracyjnego, który zawierałby spis typów datagramów i opisy wałich znaczenie. Typy datagramów ICMP są zdefiniowane w RFC-1700 (RFC *Assigned Numbers*). Są one także spisane w jednym z plików nagłówkowych standardowej biblioteki C. Typy datagramów można też znaleźć w pliku */usr/include/netinet/ip_icmp.h*, należącym do pakietu standardowej biblioteki GNU i używanym przez programistów C do pisania oprogramowania sieciowego wykorzystującego protokół ICMP. Dla twojej wygody pokazaliśmy je w tabeli 9-2. Interfejs polecenia *iptables* pozwala ci także określać typy ICMP poprzez nazwę, a więc podaliśmy także ich mnemonikę.

Tabela 9-2. Typy datagramów ICMP

Typ	*Mnemonika iptables*	*Opis typu*
0	echo-reply	Powtórzenie odpowiedzi
3	destination-unreachable	Cel nieosiągalny
4	source-quench	· ródło nieaktywne
5	redirect	Przekierowanie
8	echo-request	Żądanie powtórzenia
11	time-exceeded	Czas upłynął
12	parameter-problem	Problem z parametrem
13	timestamp-request	Żądanie znacznika czasu
14	timestamp-reply	Wysłanie znacznika czasu w odpowiedzi
15	none	Żądanie informacji
16	none	Wysłanie informacji w odpowiedzi
17	address-mask-request	Żądanie maski adresu
18	address-mask-reply	Wysłanie maski adresu w odpowiedzi

Łańcuchy firewalla IP (jądra 2.2)

Linux rozwija się, by sprostać rosnącym wymaganiom jego użytkowników. Firewall IP nie stanowi tu wyjątku. Tradycyjna implementacja firewalla IP jest dobra, ale może być niewydolna przy konfiguracji bardziej złożonych środowisk. Aby sprostać nowym wymaganiom, opracowano nową metodę konfigurowania firewalla IP i rozwinięto związane z nią właściwości. Ta nowa metoda otrzymała nazwę „Łańcuchy firewalla IP” (w skrócie łańcuchy IP) pierwszy raz została wprowadzona do użytku w jądrze Linuksa 2.2.0.

Łańcuchy IP zostały opracowane przez Paula Russella i Michaela Neulinga*. Paul udokumentował oprogramowanie łańcuchów IP w *IPCHAINS-HOWTO*.

Łańcuchy IP pozwalają na tworzenie klas reguł, do których możesz następnie dodawać hosty albo sieci lub je stamtąd usuwać. Łączenie reguł w łańcuchy jest o tyle lepsze, że poprawia wydajność firewalla w konfiguracjach, gdzie używa się wielu reguł.

Łańcuchy IP są obsługiwane przez jądra serii 2.2, ale są także dostępne w postaci łat do *jąder 2.0.**. Dokument *HOWTO* podaje, gdzie można zdobyć łaty i zawiera wiele wskazówek, jak efektywnie używać narzędzia konfiguracyjnego *ipchains*.

Używanie ipchains

Korzystać z narzędzia konfiguracyjnego *ipchains* można na dwa sposoby. Pierwszy polega na użyciu skryptu *ipfwadm-wrapper*, który w zasadzie udaje *ipfwadm*, bo wywołuje w tle program *ipchains*. Jeżeli chcesz tak używać *ipchains*, ten podrozdział nie jest ci potrzebny. Lepiej wrócić do poprzednich, opisujących *ipfwadm*, i tylko zastąpić go przez *ipfwadm-wrapper*. Skrypt ten będzie działał, ale nie ma gwarancji, że będzie utrzymywany, a poza tym nie będziesz czerpał korzyści z zaawansowanych funkcji łańcuchów IP.

Można też używać *ipchains* inaczej. Trzeba się nauczyć nowej składni i zmodyfikować istniejącą konfigurację do postaci zgodnej z nową składnią. Chwila zastanowienia i zauważysz, że w czasie konwersji jest możliwe zoptymalizowanie twojej konfiguracji. Składnia *ipchains* jest prostsza do nauczenia się niż *ipfwadm*, a więc to dobry wybór.

Program *ipfwadm* do skonfigurowania firewalla musiał operować na trzech regułach. W przypadku łańcuchów IP możesz stworzyć dowolną liczbę zestawów reguł, gdzie każda będzie połączona z inną, ale wciąż są obecne trzy zestawy reguł związane z firewallem. Standardowe zestawy reguł są bezpośrednimi odpowiednikami tych używanych w *ipfwadm*, poza tym, że mają nazwy: `input`, `forward` i `output`.

Najpierw przyjrzymy się ogólnej składni polecenia *ipchains*, a następnie zobaczymy, jak używać *ipchains* zamiast *ipfwadm*, przy czym nie uwzględnialiśmy zaawansowanych funkcji łączenia w łańcuchy. Zrobimy to, przeglądając nasze poprzednie przykłady.

Składnia polecenia ipchains

Składnia polecenia *ipchains* jest prosta. Przyjrzymy się teraz najważniejszym jej elementom. Ogólna składnia większości poleceń *ipchains* jest następująca:

```
ipchains polecenie określenie-reguły opcje
```

* Z Paulem można się skontaktować pod adresem *Paul.Russell@rustcorp.com.au*.

Polecenia

Istnieje szereg sposobów na operowanie na regułach i zestawach reguł za pomocą polecenia *ipchains*. Istotne dla firewalla IP są:

-A łańcuch
: Dodanie jednej lub kilku reguł na koniec zadanego łańcucha. Jeżeli jest podana nazwa źródłowego lub docelowego hosta i tłumaczy się na więcej niż jeden adres IP, zostanie dodana reguła dla każdego z tych adresów.

-I łańcuch numer reguły
: Wstawienie jednej lub kilku reguł na początek zadanego łańcucha. Znów, jeżeli w określeniu reguły zostanie podana nazwa hosta, będzie dodawana do każdego adresu.

-D łańcuch
: Usunięcie jednej lub kilku reguł z zadanego łańcucha, który pasuje do reguły.

-D łańcuch numer reguły
: Usunięcie reguły znajdującej się na pozycji `numerregu□y` w zadanym łańcuchu. Numeracja reguł zaczyna się od pierwszej reguły w łańcuchu.

-R łańcuch numer reguły
: Zastąpienie reguły na pozycji `numerregu□y` w zadanym łańcuchu podaną regułą.

-C łańcuch
: Sprawdzenie zadanym łańcuchem datagramu opisanego regułą. Polecenie to zwraca komunikat opisujący, jak datagram był przetwarzany przez łańcuch. Jest to bardzo użyteczna opcja to testowania konfiguracji firewalla i nieco później przyjrzymy się jej bardziej szczegółowo.

-L [łańcuch]
: Listowanie reguł zadanego łańcucha lub wszystkich łańcuchów, jeżeli żaden nie zostanie zadany.

-F [łańcuch]
: Usunięcie reguł z zadanego łańcucha lub usunięcie wszystkich reguł, jeżeli żaden łańcuch nie zostanie zadany.

-Z [łańcuch]
: Wyzerowanie liczników datagramów i bajtów dla wszystkich reguł zadanego łańcucha lub wszystkich łańcuchów, jeżeli żaden nie zostanie zadany.

-N łańcuch
: Stworzenie nowego łańcucha o zadanej nazwie. Nie może istnieć drugi łańcuch o tej samej nazwie. W ten sposób tworzone są łańcuchy definiowane przez użytkownika.

-X [łańcuch]
: Usunięcie zadanego łańcucha zdefiniowanego przez użytkownika lub wszystkich łańcuchów zdefiniowanych przez użytkownika, jeżeli żaden nie został zadany. Aby to polecenie zadziałało, nie może być odwołań do zadanego łańcucha z żadnych innych reguł.

-P łańcuch polityka

Ustawienie domyślnej polityki dla zadanego łańcucha. Dopuszczalne polityki to `ACCEPT`, `DENY`, `REJECT`, `REDIR` i `RETURN`. Polityki `ACCEPT`, `DENY` i `REJECT` mają takie samo znaczenie jak w tradycyjnych implementacjach firewalla. `REDIR` oznacza, że datagram powinien być niewidocznie przekierowany na port firewalla. `RETURN` powoduje, że kod firewalla IP powraca do tego łańcucha firewalla, który wywołał łańcuch zawierający tę regułę, i kontynuuje dalsze działanie, począwszy od następnej reguły.

Parametry określające regułę

Na regułę *ipchains* składa się wiele parametrów, które określają, jakie typy pakietów mają do niej pasować. Jeżeli którykolwiek z tych parametrów zostanie w regule pominięty, zakładana jest jego wartość domyślna.

-p [!] protokół

Określa protokół datagramu, który będzie pasował do tej reguły. Dopuszczalne nazwy protokołów to `tcp`, `udp`, `icmp` lub `all`. Możesz także podać numer protokołu. Na przykład mógłbyś użyć `4`, aby dopasować protokół enkapsulacji `ipip`. Jeżeli podasz `!`, reguła zostanie zanegowana i datagram będzie dopasowywany do wszystkich protokołów poza zadanymi. Jeżeli parametr nie zostanie podany, zostanie przyjęta wartość `all`.

-s [!]adres[/maska][!][port]

Określa adres źródłowy i port w datagramie, który ma pasować do tej reguły. Adres może być podany w postaci nazwy hosta, nazwy sieci lub adresu IP. Opcja `mask` pozwala na zadanie używanej maski sieci, która może być podana albo w tradycyjnej postaci (tj. /255.255.255.0), albo w postaci współczesnej (tj. /24).

Opcjonalny `port` określa port TCP lub UDP albo typ dopasowywanego datagramu ICMP. Numer portu możesz wskazać tylko wtedy, gdy użyłeś wcześniej parametru *–p*, podając jeden z protokołów: `tcp`, `udp` lub `icmp`. Można też podać zakres portów – jego dolną i górną granicę rozdzielone dwukropkiem. Na przykład `20:25` opisuje wszystkie porty od 20 do 25 włącznie. Znak ! może być wykorzystany do zanegowania wartości.

-d [!]adres[/maska][!][port]

Określa adres i port docelowy zawarte w datagramie, który ma pasować do tej reguły. Kodowanie tego parametru jest identyczne jak parametru *–s*.

-j cel

Określa działanie do wykonania, jeżeli reguła będzie pasowała. Parametr ten możesz sobie przetłumaczyć jako „skocz do" (ang. *jump to*). Dopuszczalne cele to `ACCEPT`, `DENY`, `REJECT`, `REDIR` i `RETURN`. Ich znaczenie opisaliśmy wcześniej. Jednak możesz podać także nazwę łańcucha zdefiniowanego przez użytkownika, w którym będzie wykonywane dalsze przetwarzanie. Jeżeli ten parametr zostanie pominięty, to nawet jeśli datagram pasuje do reguły, nie zostanie podjęte żadne inne działanie, oprócz uaktualnienia datagramu i liczników bajtów.

-i[!]nazwa-interfejsu
: Określa interfejs, który przyjął datagram lub przez który zostanie on wysłany. Znów znak ! odwraca wynik dopasowania. Jeżeli nazwa interfejsu kończy się znakiem +, pasował będzie każdy interfejs, którego nazwa rozpoczyna się zadanym ciągiem. Na przykład, `-i ppp+` będzie pasować do dowolnego urządzenia sieciowego PPP, a `-i ! eth+` będzie pasować do wszystkich urządzeń poza Ethernetem.

[!]-f
: Mówi, że reguła ta dotyczy wszystkiego poza pierwszym fragmentem datagramu podzielonego na fragmenty.

Opcje

Poniżej pokazane opcje *ipchains* są bardziej ogólne. Niektóre z nich sterują raczej ezoterycznymi funkcjami oprogramowania łańcuchów IP:

-b
: Powoduje, że polecenie generuje dwie reguły. Jedna reguła uwzględnia podane parametry, a druga uwzględnia je w odwrotnym kierunku.

-v
: Powoduje, że *ipchains* wyświetla bogate wyniki, czyli podaje więcej informacji.

-n
: Powoduje, że *ipchains* wyświetla adres IP i porty jako liczby bez próby ich zamiany na odpowiadające im nazwy.

-l
: Włącza zapisywanie przez jądro pasujących datagramów. Wszystkie datagramy pasujące do reguły są zapisywane przez jądro za pomocą funkcji *printk()*. Zwykle jest to obsługiwane przez program *syslogd* zapisujący do pliku log. Funkcja ta przydaje się do oglądania niestandardowych datagramów.

-o[maxrozmiar]
: Powoduje, że oprogramowanie łańcuchów IP kopiuje datagramy pasujące do reguły do urządzenia „netlink", które działa w przestrzeni użytkownika. Argument *maxrozmiar* ogranicza liczbę bajtów każdego datagramu, która zostanie przekazana do urządzenia netlink. Opcja ta jest najchętniej stosowana przez programistów, ale może być w przyszłości wykorzystana w pakietach oprogramowania.

-m wartośćznakowania
: Powoduje, że pasujące datagramy są *oznaczane* zadaną wartością. Te wartości to 32-bitowe liczby bez znaku. W obecnych implementacjach opcja ta nie działa, ale w przyszłości może decydować o obsłudze datagramu przez inne oprogramowanie, takie jak na przykład kod rutujący. Jeżeli *wartośćznakowania* rozpoczyna się od znaku + albo -, jest ona dodawana lub odejmowana od aktualnej wartości znakowania.

-t ma ska and ma skaxor

Po zwa la na ope ro wa nie na bi tach „typ usługi" w nagłówku IP ka ż de go da ta gramu pa sującego do reguły. Bity typu usługi są uży wa ne przez in te li gent ne ru tery do nadawania priorytetów datagramom, zanim zostaną dalej przekazane. Opro gra mo wa nie ru tujące Linuk sa po tra fi re ali zo wać ta kie nada wa nie prio ry tetów. War to ści `maskaand` i `maskaxor` ozna czają ma ski bi to we, któ re będą poddawane od po wied nio lo gicz nej ope ra cji AND i OR z bi ta mi typu usługi da tagramu. Jest to zaawansowana funkcja, która szczegółowo została omówiona w *IPCHAINS-HOWTO*.

-x

Po wo du je, że wszel kie licz by w wy niku *ipchains* są po kaz ywa ne dokład nie, bez zaokrąglania.

-y

Po wo du je, że do reguły pa sują wszyst kie da tag ramy TCP z ustaw ionym bi tem SYN i wyzerowanymi bitami ACK i FIN. Jest używana do filtrowania żądań nawiąza nia połącze nia TCP.

Po pra wio na wer sja na sze go pro ste go przykładu

Powróćmy do przykładu z sie cią firmową, w której działa fi re wall opar ty na kom pute rze z Linuk sem. Umo żli wia on użyt kow ni kom do stęp do serwerów WWW w Inter ne cie, ale nie po zwa la na ża den in ny ruch.

Gdy by na sza sieć miała 24- bitową ma skę (kla sa C) i ad res 172.16.1.0, użyl iby śmy następ ujących re guł *ipchains*:

```
# ipchains -F forward
# ipchains -P forward DENY
# ipchains -A forward -s 0/0 80 -d 172.16.1.0/24 -p tcp -y -j DENY
# ipchains -A forward -s 172.16.1.0/24 -d 0/0 80 -p tcp -b -j ACCEPT
```

Pierw sze po le ce nie czy ści wszyst kie re guły z ze sta wu `forward`, a dru gie de fi niuje domyślną politykę zestawu re guł `forward` na `DENY`. Trze cie i czwar te po le cenie re ali zują wy ma ga ne przez nas fil tro wa nie. Czwar te po le ce nie po zwa la da ta gramom kie ro wa nym do i z ser we rów WWW na prze cho dze nie do na szej sie ci, a trzecie zapobiega przyjmowaniu przychodzących połączeń TCP z portem źródłowym 80.

Gdybyśmy te raz chcie li do dać re guły po zwa lające na do stęp do ze w nętrz nych serwerów FTP w try bie bier nym, mu sie li by śmy wpi sać:

```
# ipchains -A forward -s 0/0 20 -d 172.16.1.0/24 -p tcp -y -j DENY
# ipchains -A forward -s 172.16.1.0/24 -d 0/0 20 -p tcp -b -j ACCEPT
# ipchains -A forward -s 0/0 21 -d 172.16.1.0/24 -p tcp -y -j DENY
# ipchains -A forward -s 172.16.1.0/24 -d 0/0 21 -p tcp -b -j ACCEPT
```

Listowanie reguł za pomocą ipchains

Do wylistowania reguł za pomocą *ipchains* używamy argumentu *–L*. Podobnie jak to było w *ipfwadm*, istnieją argumenty kontrolujące liczbę szczegółów pokazywanych w wyniku. W najprostszej postaci *ipchains* generuje następujący wynik:

```
# ipchains -L -n
Chain input (policy ACCEPT):
Chain forward (policy DENY):
target     prot  opt     source            destination       ports
DENY       tcp   -y----  0.0.0.0/0         172.16.1.0/24     80 -> *
ACCEPT     tcp   ------  172.16.1.0/24     0.0.0.0/0         * -> 80
ACCEPT     tcp   ------  0.0.0.0/0         172.16.1.0/24     80 -> *
ACCEPT     tcp   ------  172.16.1.0/24     0.0.0.0/0         * -> 20
ACCEPT     tcp   ------  0.0.0.0/0         172.16.1.0/24     20 -> *
ACCEPT     tcp   ------  172.16.1.0/24     0.0.0.0/0         * -> 21
ACCEPT     tcp   ------  0.0.0.0/0         172.16.1.0/24     21 -> *

Chain output (policy ACCEPT):
```

Jeżeli nie podasz nazwy łańcucha, który chcesz obejrzeć, *ipchains* wyświetli wszystkie reguły ze wszystkich łańcuchów. Argument `-n` w naszym przykładzie powoduje, że *ipchains* nie próbuje konwertować adresów i portów na nazwy. Pokazana informacja powinna być oczywista.

Bogatsza forma wyniku, uzyskiwana przez opcję *–u*, pokazuje dużo więcej szczegółów. Dodatkowe pola zawierają liczniki datagramów i bajtów, znaczniki *AND* i *XOR* typu usługi, nazwę interfejsu, znakowanie i rozmiar wynikowy.

Ze wszystkimi regułami utworzonymi za pomocą *ipchains* związane są liczniki bajtów i datagramów. W ten sposób jest zaimplementowane liczenie ruchu IP, które zostanie szczegółowo omówione w rozdziale 10. Domyślnie liczniki są pokazywane w postaci zaokrąglonej z przyrostkami `K` i `M` oznaczającymi odpowiednio jednostki: tysiąc i milion. Jeżeli zostanie podany argument `-x`, liczniki są rozwijane do ich pełnej, niezaokrąglonej postaci.

Korzystanie z łańcuchów

Wiesz już, że polecenie *ipchains* zastępuje *ipfwadm*, ma prostszą składnię i kilka ciekawych rozszerzeń, ale bez wątpienia chcesz wiedzieć, gdzie i po co używać łańcuchów definiowanych przez użytkownika. Zapewne jesteś też ciekawy, jak posługiwać się dodatkowymi skryptami towarzyszącymi poleceniu *ipchains* w pakiecie. Przyjrzymy się teraz tym tematom i postaramy się odpowiedzieć na pytania.

Łańcuchy definiowane przez użytkownika

Trzy zestawy reguł dla tradycyjnego firewalla IP stanowią mechanizm tworzenia prostych konfiguracji firewalla, którymi łatwo jest zarządzać w małych sieciach o niewielkich wymaganiach wobec systemu bezpieczeństwa. Gdy wymagania konfiguracyjne wzrastają, pojawia się szereg problemów. Po pierwsze, duże sieci często wymagają dużo więcej reguł firewalla, niż tych kilka, z którymi się do tej pory spotkaliśmy. Nieuchronnie rosną potrzeby dodawania do firewalla reguł obsługujących przypadki szczególne. Gdy liczba reguł rośnie, wydajność firewalla pogarsza się, bo

na każdym datagramie jest przeprowadzanych coraz więcej testów; problemem staje się też zarządzanie. Po drugie, nie jest możliwe włączanie i wyłączanie zestawu reguł w sposób rozdzielny. Gdy jesteś w trakcie przebudowy zestawu reguł, narażasz sieć na ataki.

Zasady budowy łańcuchów IP pomagają złagodzić te problemy, gdyż umożliwiają administratorowi tworzenie dowolnych zestawów reguł firewalla, które można następnie dołączać do trzech wbudowanych zestawów reguł. Do utworzenia nowego łańcucha można użyć opcji *–N* programu *ipchains*. Trzeba podać jego nazwę, składającą się z 8 (lub mniej) znaków. (Ograniczenie nazwy do małych liter jest dobrym pomysłem). Opcja *–j* konfiguruje działanie podejmowane wtedy, gdy datagram pasuje do wymagań reguły. Mówi, że jeżeli datagram będzie pasował do reguły, dalsze testowanie powinno być realizowane w łańcuchu zdefiniowanym przez użytkownika. Pokażemy to na wykresie.

Rozważmy następujące polecenia *ipchains*:

```
ipchains -P input DENY
ipchains -N tcpin
ipchains -A tcpin -s ! 172.16.0.0/16
ipchains -A tcpin -p tcp -d 172.16.0.0/16 ssh -j ACCEPT
ipchains -A tcpin -p tcp -d 172.16.0.0/16 www -j ACCEPT
ipchains -A input -p tcp -j tcpin
ipchains -A input -p all
```

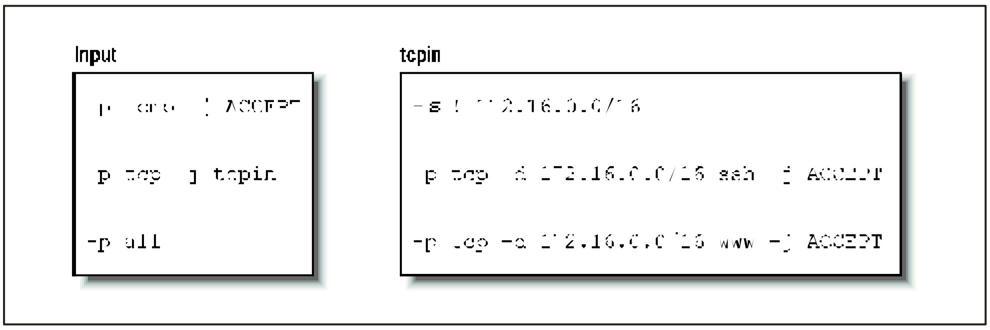
Domyślną politykę łańcucha wejściowego ustawiamy na `deny`. Drugie polecenie tworzy zdefiniowany przez użytkownika łańcuch o nazwie „tcpin". Trzecie polecenie dodaje do łańcucha `tcpin` regułę, do której pasują wszystkie datagramy pochodzące spoza naszej sieci lokalnej. Nie jest podejmowane żadne dodatkowe działanie. Jest to reguła zliczająca i zostanie omówiona bardziej szczegółowo w rozdziale 10. Do następnych dwóch reguł pasują datagramy przeznaczone dla naszej sieci lokalnej na porty `ssh` lub `www`. Pasujące datagramy są akceptowane. W następnej regule tkwi prawdziwa magia *ipchains*. Reguła ta powoduje, że oprogramowanie firewalla sprawdza każdy datagram TCP za pomocą łańcucha `tcpin`, zdefiniowanego przez użytkownika. Na koniec dodajemy regułę do naszego łańcucha `input`, do którego pasuje każdy datagram. Jest to kolejna reguła zliczająca. W ten sposób uzyskujemy łańcuchy firewalla pokazane na rysunku 9-4.

Nasze łańcuchy `input` i `tcpin` są zapełniane regułami. Przetwarzanie datagramu zawsze rozpoczyna się w jednym z łańcuchów wbudowanych. Zobaczymy, jak zdefiniowany przez nas łańcuch jest używany przy przetwarzaniu różnych typów datagramów.

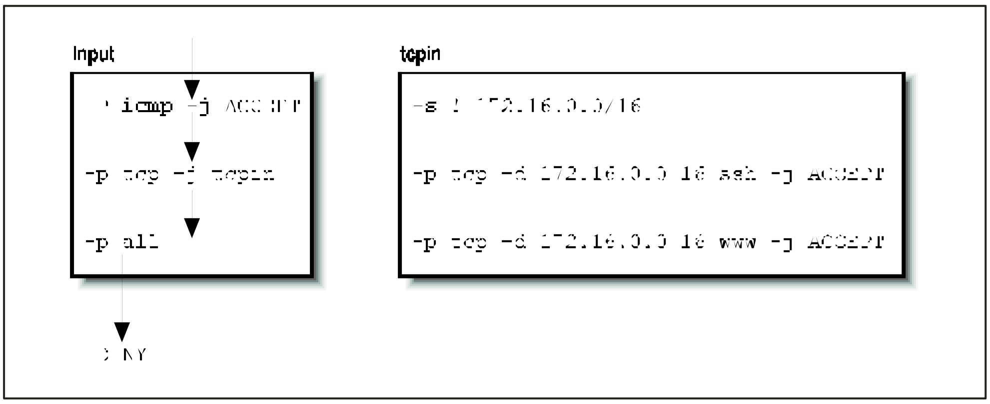
Najpierw przyjrzymy się, co się dzieje, gdy zostanie odebrany datagram UDP dla jednego z naszych hostów. Rysunek 9-5 pokazuje przepływ przez reguły.

Datagram zostaje odebrany przez łańcuch `input`, ale nie pasuje do dwóch pierwszych reguł, ponieważ pasują do nich tylko datagramy protokołów ICMP i TCP. Zostaje dopasowany do trzeciej reguły łańcucha `input`, która nie zawiera celu. Są więc uaktualniane liczniki bajtowy i datagramów, ale nie jest podejmowane żadne inne

działanie. Datagram dociera do końca łańcucha `input`, spełniając warunki jego domyślnej polityki i zostaje odrzucony.

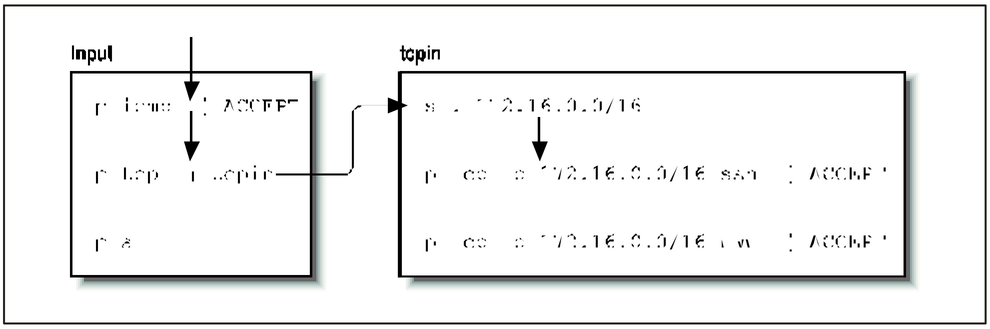


Rysunek 9-4. Prosty zestaw reguł łańcucha IP



Rysunek 9-5. Kolejność sprawdzania reguł dla odebranego datagramu UDP

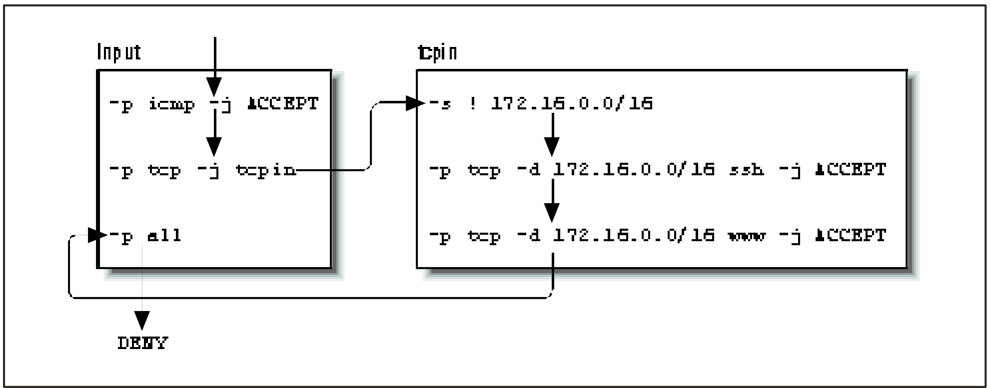
Aby zobaczyć zdefiniowany przez nas łańcuch w działaniu, rozważmy, co się dzieje, gdy zostanie odebrany datagram TCP przeznaczony dla portu `ssh` jednego z naszych hostów. Kolejność pokazano na rysunku 9-6.



Rysunek 9-6. Kolejność reguł dla odebranego pakietu TCP dla portu ssh

Tym razem datagram pasuje do drugiej reguły w łańcuchu `input`, która kieruje go do celu `tcpin` – łańcucha zdefiniowanego przez użytkownika. Podanie łańcucha zdefiniowanego przez użytkownika jako celu powoduje, że datagram będzie sprawdzany przez zawarte w nim reguły, a więc następną sprawdzaną regułą będzie pierwsza reguła z łańcucha `tcpin`. Do tej reguły pasują datagramy, które mają adres źródłowy spoza sieci lokalnej i nie zawierają adresu przeznaczenia, a więc jest to także reguła zliczająca i testowanie przechodzi do następnej reguły. Druga reguła w naszym łańcuchu `tcpin` pasuje do datagramu i określa cel `ACCEPT`. Dotarliśmy do celu, a więc nie będzie już żadnego przetwarzania przez firewall. Datagram zostaje przepuszczony.

Na koniec zobaczmy, co się stanie, gdy dotrzemy do końca zdefiniowanego przez nas łańcucha. Aby to zobaczyć, musimy pokazać przepływ datagramu TCP przeznaczonego dla portu innego niż dwa przez nas obsługiwane. Pokazujemy to na rysunku 9-7.



Rysunek 9-7. Kolejność reguł dla odebranego pakietu TCP dla telnet

Łańcuchy zdefiniowane przez użytkownika nie mają polityki domyślnej. Gdy wszystkie zawarte w nich reguły zostaną sprawdzone i żadna nie pasuje, firewall działa tak, jakby istniała reguła `RETURN`, a więc jeżeli nie tego chciałeś, powinieneś na końcu łańcucha użytkownika umieścić żądane działanie. W naszym przykładzie sprawdzanie powraca do reguły z zestawu `input` następnej po tej, przez którą przeszliśmy do łańcucha zdefiniowanego przez użytkownika. Ostatecznie docieramy do końca łańcucha `input`, który posiada politykę domyślną i datagram zostaje odrzucony.

Ten przykład jest bardzo prosty, ale pokazuje to, o co nam chodziło. W praktyce działanie łańcuchów IP jest dużo bardziej skomplikowane. Nieco bardziej wyrafinowany przykład pokazujemy poniżej, w postaci listy poleceń.

```
#
# Ustawienie domy□lnej polityki przekazywania na REJECT
ipchains -P forward REJECT
#
# utworzenie naszego □a□cucha
ipchains -N sshin
```

```
ipchains -N sshout
ipchains -N wwwin
ipchains -N wwwout
#
# Gwarancja odrzucania po□□cze□ w z□ym kierunku
ipchains -A wwwin -p tcp -s 172.16.0.0/16 -y -j REJECT
ipchains -A wwwout -p tcp -d 172.16.0.0/16 -y -j REJECT
ipchains -A sshin -p tcp -s 172.16.0.0/16 -y -j REJECT
ipchains -A sshout -p tcp -d 172.16.0.0/16 -y -j REJECT
#
# Gwarancja, □e wszystko, co dotrze do ko□ca □a□cucha
# zdefiniowanego przez u□ytkownika, zostanie odrzucone
ipchains -A sshin -j REJECT
ipchains -A sshout -j REJECT
ipchains -A wwwin -j REJECT
ipchains -A wwwout -j REJECT
#
# Przekierowanie us□ug www i ssh do odpowiedniego □a□cucha
# zdefiniowanego przez u□ytkownika
ipchains -A forward -p tcp -d 172.16.0.0/16 ssh -b -j sshin
ipchains -A forward -p tcp -s 172.16.0.0/16 -d 0/0 ssh -b -j sshout
ipchains -A forward -p tcp -d 172.16.0.0/16 www -b -j wwwin
ipchains -A forward -p tcp -s 172.16.0.0/16 -d 0/0 www -b -j wwwout
#
# Umieszczenie regu□ sprawdzaj□cych hosty na drugiej pozycji w
# zdefiniowanych przez nas □a□cuchach
ipchains -I wwwin 2 -d 172.16.1.2 -b -j ACCEPT
ipchains -I wwwout 2 -s 172.16.1.0/24 -b -j ACCEPT
ipchains -I sshin 2 -d 172.16.1.4 -b -j ACCEPT
ipchains -I sshout 2 -s 172.16.1.4 -b -j ACCEPT
ipchains -I sshout 2 -s 172.16.1.6 -b -j ACCEPT
#
```

W tym przykładzie użyliśmy łańcuchów definiowanych przez użytkownika do uproszczenia zarządzania naszym firewallem i do poprawy wydajności w porównaniu z rozwiązaniem wykorzystującym jedynie łańcuchy wbudowane.

W naszym przykładzie są tworzone łańcuchy użytkownika dla każdej z usług `ssh` i `www` w każdym kierunku połączenia. W łańcuchu `wwwout` umieszczamy reguły dla hostów, które mogą tworzyć wychodzące połączenia WWW, a w `sshin` definiujemy reguły dla hostów, które mogą przyjmować przychodzące połączenia ssh. Założyliśmy, że potrzebujemy możliwości przyjmowania i odrzucania połączeń `ssh` i `www` tylko dla wybranych hostów w naszej sieci. Jest to pewne uproszczenie, gdyż łańcuchy definiowane przez użytkownika pozwalają na grupowanie reguł według pakietów przychodzących do hosta i wychodzących z niego, a nie na ich mieszanie. Poprawia się wydajność, ponieważ dla każdego datagramu zmniejszyliśmy średnią liczbę testów robionych przed osiągnięciem celu. Wydajność zwiększy się jeszcze bardziej, jeżeli wzrośnie liczba hostów. Gdybyśmy nie mieli łańcuchów użytkownika, potencjalnie musielibyśmy przeszukiwać całą listę reguł, by stwierdzić, czy działanie ma zostać podjęte przy każdym odebranym datagramie. Nawet gdybyśmy założyli, że każda z reguł zawartych na naszej liście pasuje do równej liczby przetworzonych datagramów, wciąż musielibyśmy przeszukiwać średnio połowę listy. Łańcuchy definiowane przez użytkownika pozwalają nam uniknąć sprawdza-

nia dużej liczby reguł, ponieważ testowany datagram musi pasować do prostej reguły wbudowanego łańcucha, aby w ogóle dotrzeć do łańcucha użytkownika.

Skrypty pomocnicze ipchains

Pakiet oprogramowania *ipchains* jest dostarczany wraz z trzema dodatkowymi skryptami. Pierwszy z nich już krótko omówiliśmy, natomiast pozostałe dwa zapewniają łatwe i wygodne sposoby zachowywania i odtwarzania konfiguracji firewalla.

Skrypt *ipfwadm-wrapper* emuluje składnię wiersza poleceń *ipfwadm*, ale wymaga polecenia *ipchains* do tworzenia reguł. Jest to wygodny sposób na migrację istniejącej konfiguracji firewalla do jądra lub alternatywa dla opanowania składni *ipchains*. Skrypt *ipfwadm-wrapper* zachowuje się inaczej niż polecenie *ipfwadm* pod dwoma względami. Po pierwsze, *ipchains* nie pozwala na określenie interfejsu przez adres, a *ipfwadm-wrapper* przyjmuje argument *–V*, ale próbuje zamienić go na właściwy dla *ipchains* odpowiednik *–W*, szukając nazwy interfejsu skonfigurowanej pod zadanym adresem. Skrypt *ipfwadm-wrapper* zawsze przypomina o tym, wypisując komunikat, gdy użyjesz opcji *–V*. Po drugie, reguły zliczania fragmentów nie są tłumaczone poprawnie.

Skrypty *ipchains-save* i *ipchains-restore* upraszczają tworzenie i modyfikowanie konfiguracji firewalla. Polecenie *ipchains-save* odczytuje aktualną konfigurację firewalla i zapisuje uproszczoną postać na standardowe wyjście. Polecenie *ipchains-restore* odczytuje dane w formacie wyprowadzanym przez *ipchains-save* i konfiguruje firewall IP zgodnie z odczytanymi regułami. Korzyścią z używania tych skryptów jest możliwość natychmiastowego dynamicznego tworzenia konfiguracji i jej zapisania, czego nie daje bezpośrednie modyfikowanie skryptu konfigurującego firewall i testowanie konfiguracji. Taką konfigurację można następnie odtworzyć, zmodyfikować i zapisać ponownie.

Aby użyć tych skryptów i zachować aktualną konfigurację firewalla, musisz napisać coś takiego:

```
ipchains-save >/var/state/ipchains/firewall.state
```

Możesz ją potem odtworzyć, zwykle w czasie uruchamiania systemu, w następujący sposób:

```
ipchains-restore </var/state/ipchains/firewall.state
```

Skrypt *ipchains-restore* sprawdza, czy istnieją już umieszczone w konfiguracji łańcuchy zdefiniowane przez użytkownika. Jeśli podasz opcję `-f`, reguły z wcześniej skonfigurowanych łańcuchów użytkownika będą automatycznie usunięte przed wczytaniem nowych. Natomiast przy działaniu domyślnym jesteś pytany, czy pozostawić, czy usunąć dany łańcuch.

Netfilter i tabele IP (jądra 2.4)

Kiedy Paul Russell pracował nad łańcuchami IP, doszedł do wniosku, że firewalle IP powinny być mniej skomplikowane. Wkrótce więc zajął się upraszczaniem przetwarzania datagramów w kodzie firewalla zawartym w jądrze i stworzył strukturę

filtrującą, która była dużo prostsza i dużo bardziej elastyczna. Tę nową strukturę nazwał *netfilter*.



W czasie pisania tej książki budowa *netfilter* nie była jeszcze ustabilizowana. Mamy nadzieję, że wybaczysz nam wszelkie błędy w opisie *netfilter* i narzędzi konfiguracyjnych, które wynikają ze zmian, jakie zaszły po przygotowaniu tego materiału. Uznaliśmy, że *netfilter* jest tematem na tyle istotnym, by usprawiedliwić umieszczenie jego roboczego opisu w tej książce, bez względu na fakt, że jego części są oparte na domysłach. Gdy byś miał jakiekolwiek wątpliwości, są już dostępne odpowiednie dokumenty HOWTO zawierające bardziej dokładne i aktualne informacje na temat szczegółowych zagadnień związanych z konfiguracją *netfilter*.

Cóż więc było nie tak z łańcuchami IP? Poprawiły one znacznie wydajność i zarządzanie regułami firewalla. Ale sposób przetwarzania datagramów wciąż był skomplikowany, szczególnie w połączeniu z funkcjami związanymi z firewallem, takimi jak maskowanie IP (omówione w rozdziale 11) i inne formy translacji adresów. Częściowa złożoność wynikała z faktu, że maskowanie IP i translacja adresów sieciowych powstały niezależnie od kodu firewalla w jądrze i dopiero później zostały do niego dołączone. Niestety nie było to wszystko tworzone razem od początku i zintegrowane w kodzie firewalla. Gdyby twórcy chcieli dodać następne funkcje związane z przetwarzaniem datagramów, mieliby trudności ze znalezieniem miejsca na umieszczenie swojego kodu i musieliby dokonać zmian w jądrze, aby dopiąć swego.

Poza tym istniały jeszcze inne problemy. W szczególności łańcuch „input" opisywał wejście do całej warstwy sieci IP. Łańcuch wejściowy dotyczył zarówno datagramów *przeznaczonych dla* hosta, jak i datagramów *rutowanych przez* host. Było to nieco mylące, ponieważ mieszało funkcję łańcucha wejściowego z funkcją łańcucha przekazującego, dotyczącego tylko datagramów przekazywanych dalej, ale zawsze sprawdzanych po przejściu przez łańcuch wejściowy. Gdybyś chciał inaczej traktować datagramy przeznaczone dla hosta niż datagramy przekazywane dalej, musiałbyś stworzyć złożone reguły wykluczające jedne lub drugie. Ten sam problem dotyczył łańcucha wyjściowego.

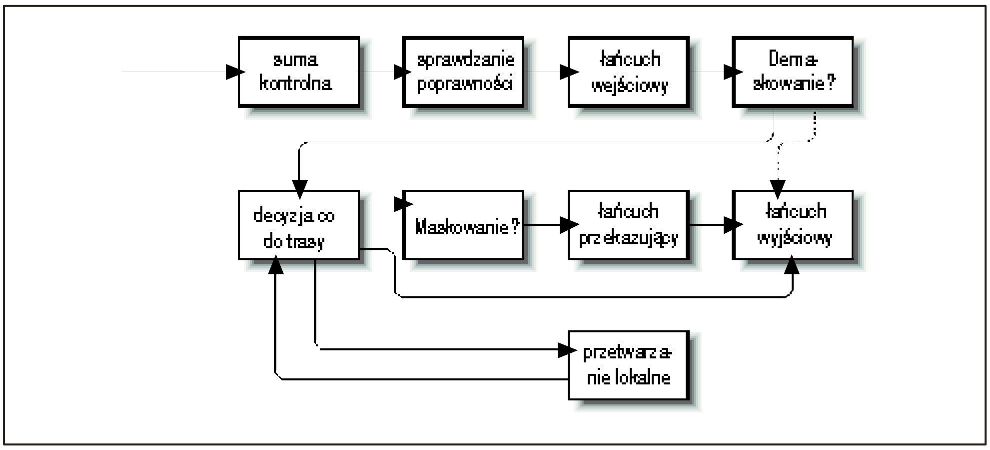
Oczywiście ta złożoność poniekąd dotknęła administratora systemu, ponieważ odbiła się na sposobie tworzenia reguł. Co więcej, wszelkie rozszerzenia filtrowania wymagały bezpośrednich modyfikacji jądra, ponieważ wszystkie polityki filtrowania były w nim zaimplementowane i nie było sposobu na stworzenie przezroczystego interfejsu. *netfilter* radzi sobie zarówno ze złożonością, jak i sztywnością starszych rozwiązań, implementując w jądrze ogólną strukturę określającą sposób przetwarzania datagramów i zapewniającą możliwość rozbudowy polityki filtrowania bez potrzeby modyfikacji jądra.

Przyjrzyjmy się dwóm kluczowym zmianom, które zostały dokonane. Rysunek 9-8 pokazuje, jak datagramy są przetwarzane w implementacji łańcuchów IP, natomiast rysunek 9-9 pokazuje, jak są one przetwarzane przez *netfilter*. Zasadnicze różnice to usunięcie z głównego kodu funkcji maskowania i zmiana umiejscowienia łańcu-

chów wejściowego i wyjściowego. Zmianom tym towarzyszy nowe, dające się rozbudować narzędzie o nazwie *iptables*.

W łańcuchach IP łańcuch wejściowy dotyczył wszystkich datagramów odebranych przez host bez względu na to, czy były one dla niego przeznaczone, czy rutowane do innego hosta. W *netfilter* łańcuch wejściowy dotyczy *tylko* datagramów przeznaczonych dla hosta lokalnego, a łańcuch przekazujący dotyczy tylko datagramów przeznaczonych dla *innego* hosta. Podobnie w łańcuchach IP, łańcuch wyjściowy dotyczy wszystkich datagramów wychodzących z hosta lokalnego bez względu na to, czy są to datagramy na nim stworzone, czy przez niego rutowane z innego hosta. W *netfilter* łańcuch wyjściowy dotyczy *tylko* datagramów wygenerowanych na danym hoście, a nie dotyczy datagramów przez niego rutowanych. Sama ta zmiana stanowi poważne uproszczenie wielu konfiguracji firewalla.

Na rysunku 9-8 elementy opisane jako „demaskowanie" i „maskowanie" są oddzielnymi elementami jądra od powiedzialnymi za przetwarzanie przychodzących i wychodzących datagramów maskowanych. Zostały one ponownie zaimplementowane w *netfilter* jako moduły.



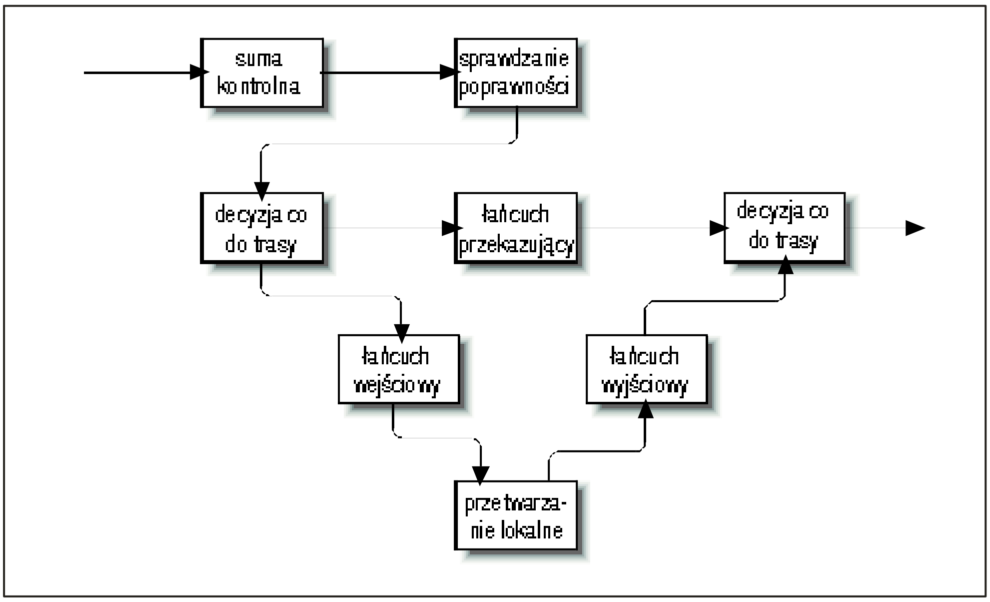
Rysunek 9-8. Łańcuch przetwarzania datagramów w łańcuchach IP

Przyjrzyjmy się konfiguracji, w której domyślna polityka dla każdego łańcucha: wejściowego, wyjściowego i przekazującego, jest ustawiona na `deny`. W łańcuchach IP potrzebne jest sześć reguł, aby przepuszczać jakąkolwiek sesję przez firewall: po dwie w każdym łańcuchu: wejściowym, wyjściowym i przekazującym (jedna obsługiwałaby przesyłanie w jedną stronę, a druga w drugą.). Możesz sobie wyobrazić, jak łatwo mogłoby to się stać nadzwyczaj skomplikowane i trudne do ogarnięcia, gdybyś chciał mieszać sesje rutowane i sesje połączeń do hosta bez rutowania. Łańcuchy IP pozwalają na tworzenie łańcuchów nieco upraszczających to zadanie, ale ich budowanie jest oczywiście i wymaga pewnego doświadczenia.

W implementacji *netfilter* ta złożoność znika zupełnie dzięki *iptables*. W przypadku usługi rutowanej przez firewalla, ale nie kończącej się na hoście lokalnym, po-

trzebne są tylko dwie reguły w łańcuchu przekazującym: jedna w jednym kierunku i druga w przeciwnym. Jest to oczywisty sposób tworzenia reguł dla firewalla i znacznie upraszcza jego konfigurację.

W dokumencie *PACKETE-FILTERING-HOWTO* znajdziesz szczegółową listę zmian, dokonywanych w czasie tworzenia oprogramowania, a więc tam szukaj bardziej praktycznych zagadnień związanych z tym tematem.



Rysunek 9-9. Łańcuch przetwarzania datagramów w netfilter

Wsteczna zgodność z ipfwadm i ipchains

Niezwykła elastyczność *netfilter* w Linuksie uwidacznia się w możliwości emulowania interfejsów *ipfwadm* i *ipchains*. Dzięki emulacji przejście na oprogramowanie firewalla nowej generacji jest dużo prostsze.

Dwa moduły jądra *netfilter*, zwane *ipfwadm.o* i *ipchains.o*, zapewniają kompatybilność wsteczną dla *ipfwadm* i *ipchains*. Możesz załadować tylko jeden z nich na raz i używać tylko wtedy, gdy moduł *ip_tables.o* nie jest załadowany. Gdy odpowiedni moduł zostanie załadowany, *netfilter* działa dokładnie tak jak poprzednia implementacja firewalla.

netfilter naśladuje interfejs *ipchains* po wydaniu następujących poleceń:

```
rmmod ip_tables
modprobe ipchains
ipchains …
```

Używanie iptables

Program *iptables* jest używany do konfigurowania reguł filtrowania *netfilter*. Jego składnia ma wiele wspólnego z *ipchains*, ale różni się pod jednym bardzo szczególnym względem: jest *rozszerzalna*. Oznacza to, że jej funkcjonalność można rozszerzyć bez ponownej kompilacji. Służą do tego biblioteki dzielone. Istnieją standardowe rozszerzenia, które omówimy za chwilę.

Zanim będziesz mógł używać polecenia *iptables*, musisz załadować moduł jądra *netfilter* niezbędny do jego obsługi. Najprościej zrobisz to za pomocą polecenia *modprobe*:

```
modprobe ip_tables
```

Polecenie *iptables* jest używane do konfigurowania zarówno filtrowania IP, jak i translacji adresów sieciowych (*Network Address Translation*). Aby temu sprostać, istnieją dwie tablice reguł: *filter* i *net*. Tablica *filter* jest używana domyślnie, jeżeli nie podasz opcji *–t* zmieniającej to zachowanie. W *netfilter* udostępniono pięć wbudowanych łańcuchów. Łańcuchy `INPUT` i `FORWARD` są dostępne dla tablicy *filter*, a `PREROUTING` i `POSTROUTING` są dostępne dla tablicy `nat`, natomiast łańcuch `OUTPUT` jest dostępny dla obu tablic. W tym rozdziale omówimy tylko tablicę *filter*. Tablicom *net* przyjrzymy się w rozdziale 11.

Ogólna składnia większości poleceń *iptables* jest następująca:

```
iptables polecenie określenie-reguły rozszerzenia
```

Teraz przyjrzymy się szczegółowo kilku opcjom, po czym podamy przykłady.

Polecenia

Istnieje szereg sposobów operowania na regułach i ich zestawach za pomocą polecenia *iptables*. Istotne dla filtrowania IP są następujące:

-A łańcuch
: Dodanie jednej lub kilku reguł na koniec zadanego łańcucha. Jeżeli zostanie podana nazwa hosta źródłowego lub docelowego, z którą związany jest więcej niż jeden adres IP, reguła zostanie dodana do każdego adresu.

-I łańcuch num_reguły
: Wstawienie jednej lub kilku reguł na początku zadanego łańcucha. Znów, jeżeli w opisie reguły zostanie podana nazwa hosta, reguła będzie dodana dla każdego adresu IP odpowiadającego temu hostowi.

-D łańcuch
: Usunięcie jednej lub kilku reguł z zadanego łańcucha, który takie reguły zawiera.

-D łańcuch num_reguły
: Usunięcie reguły znajdującej się na pozycji `num_reguły` w zadanym łańcuchu. Liczenie reguł zaczyna się od 1 w przypadku pierwszej reguły w łańcuchu.

-R łańcuch num_reguły
: Zastąpienie reguły na pozycji `num_reguły` w zadanym łańcuchu regułą o podanej charakterystyce.

-C łańcuch

Sprawdzenie zadanym łańcuchem datagramu opisanego przez regułę. To polecenie zwróci komunikat opisujący, w jaki sposób łańcuch przetworzył datagram. Jest bardzo przydatne do testowania konfiguracji firewalla i za chwilę przyjrzymy się mu bardziej szczegółowo.

-L[łańcuch]

Wylistowanie reguł z zadanego łańcucha lub ze wszystkich łańcuchów, jeżeli żaden nie zostanie wybrany.

-F [łańcuch]

Usunięcie reguł z zadanego łańcucha lub ze wszystkich łańcuchów, jeżeli żaden nie zostanie wybrany.

-Z [łańcuch]

Wyzerowanie liczników bajtów i datagramów dla wszystkich reguł w zadanym łańcuchu lub we wszystkich łańcuchach, jeżeli żaden nie zostanie wybrany.

-N łańcuch

Utworzenie nowego łańcucha o zadanej nazwie. Nie mogą istnieć łańcuchy o tej samej nazwie. W ten sposób tworzy się łańcuch definiowany przez użytkownika.

-X [łańcuch]

Usunięcie zadanego łańcucha zdefiniowanego przez użytkownika lub wszystkich takich łańcuchów, jeżeli żaden nie zostanie wybrany. Aby to polecenie zadziałało, nie może być odwołań do usuwanego łańcucha z żadnych innych łańcuchów reguł.

-P łańcuch polityka

Ustawienie domyślnej polityki dla zadanego łańcucha. Dopuszczalne polityki to `ACCEPT`, `DROP`, `QUEUE` i `RETURN`. `ACCEPT` pozwala na przepuszczenie datagramu. `DROP` powoduje, że datagram jest odrzucany. `QUEUE` powoduje, że datagram jest przekazywany do przestrzeni użytkownika w celu dalszego przetwarzania. `RETURN` powoduje, że kod firewalla IP wraca do łańcucha, który go wywołał, i kontynuuje działanie od następnej reguły.

Parametry definicji reguły

Istnieje kilka parametrów *iptables* używanych do definiowania reguły. Gdy wymagane jest zdefiniowanie reguły, musi zostać podana wartość każdego z nich albo zostaną przyjęte wartości domyślne.

-p[!]protokół

Określa protokół datagramu, który ma pasować do tej reguły. Dopuszczalne nazwy protokołów to: `tcp`, `udp`, `icmp` lub numer, jeżeli znasz numery protokołu IP*. Na przykład mógłbyś użyć liczby `4` do określenia enkapsulacji `ipip`. Gdybyś podał znak `!`, reguła zostałaby zanegowana, a datagram pasowałby do każdego protokołu poza podanym. Gdy ten parametr nie zostanie określony, domyślnie przyjęte będą wszystkie protokoły.

---

* Nazwy i numery protokołów znajdziesz w pliku */etc/protocols*.

-s[!]adres[/maska]
: Określa adres źródłowy datagramu, który będzie pasował do tej reguły. Adres może być podany w postaci nazwy hosta, nazwy sieci lub adresu IP. Opcjonalny parametr `maska` definiuje maskę sieci, która ma być zastosowana. Może być ona podana w postaci tradycyjnej (tj. /255.255.255.0) lub w postaci współczesnej (tj. /24).

-d[!]adres[/maska]
: Określa adres przeznaczenia i port datagramu, który będzie pasował do tej reguły. Kodowanie tego parametru jest takie samo jak parametru *–s*.

-j cel
: Określa, jakie działanie ma zostać podjęte, gdy reguła zostanie dopasowana. Parametr ten możesz sobie przetłumaczyć jako „skocz do" (ang. *jump to*). Dopuszczalne cele to `ACCEPT`, `DROP`, `QUEUE` i `RETURN`. Ich znaczenie opisaliśmy wcześniej w sekcji „Polecenia". Jednak możesz podać także nazwę łańcucha zdefiniowanego przez użytkownika łańcucha, w którym będzie wykonywane dalsze przetwarzanie. Możesz także podać cel obsługiwany przez rozszerzenie. Wkrótce opiszemy rozszerzenia. Jeżeli ten parametr zostanie pominięty, to jeśli datagram pasuje do reguły, nie zostanie podjęte żadne inne działanie oprócz uaktualnienia datagramu i liczników bajtów.

-i[!]nazwa-interfejsu
: Określa interfejs, który przyjął datagram. Znów znak ! odwraca wynik dopasowania. Jeżeli nazwa interfejsu kończy się znakiem +, pasował będzie każdy interfejs, którego nazwa rozpoczyna się za danym ciągiem. Na przykład, `-i ppp+` będzie pasować do dowolnego urządzenia sieciowego PPP, a `-i ! eth+` będzie pasować do wszystkich urządzeń poza Ethernetem.

-o[!]nazwa-interfejsu
: Określa interfejs, na który datagram będzie wysłany. Ten argument ma taką samą składnię jak *–i*.

[!]-f
: Mówi, że reguła ta dotyczy tylko drugiego i dalszych fragmentów datagramu. Nie dotyczy pierwszego fragmentu.

Opcje

Poniżej pokazano bardziej ogólne opcje *iptables*. Niektóre z nich sterują raczej ezoterycznymi funkcjami oprogramowania *netfilter*.

-v
: Powoduje, że *iptables* wyświetla bogate wyniki. Podawane będzie więcej informacji.

-n
: Powoduje, że *iptables* wyświetla adres IP i porty tylko jako liczby, nie próbuje zamieniać ich na odpowiadające im nazwy.

-x
: Powoduje, że wszelkie liczby w wyniku *iptables* są pokazywane dokładnie, bez zaokrąglania.

- *-numery_wierszy*

  Powoduje, że przy wyświetlaniu zestawów reguł pokazywane są numery wierszy. Numer wiersza odpowiada pozycji reguły w łańcuchu.

Rozszerzenia

Powiedzieliśmy wcześniej, że *iptables* jest narzędziem rozszerzalnym poprzez opcjonalne moduły bibliotek dzielonych. Istnieją standardowe rozszerzenia udostępniające funkcje *ipchains*. Aby z nich skorzystać, musisz podać po lecceniu *iptables* ich nazwę poprzez argument `-m nazwa`. Poniższa lista pokazuje opcje `-m` i `-p` określające kontekst rozszerzeń oraz opcje udostępniane przez rozszerzenie.

Rozszerzenia TCP: używane z –m tcp –p tcp

- *-sport [!][port[:port]]*

  Określa port, z którego musi pochodzić datagram, aby pasował do reguły. Można wyznaczyć pewien zakres portów, wypisując górny i dolny limit (należy rozdzielić je dwukropkiem). Na przykład `20:25` oznacza wszystkie porty o numerach od 20 do 25 włącznie. Znów znak `!` może być użyty do zanegowania wartości.

- *-dport[!][port[:port]]*

  Określa port, do którego musi być skierowany datagram, aby pasował do reguły. Argument jest kodowany identycznie jak *--sport*.

- *-tcp-flags [!] maska lista*

  Określa, że reguła pasuje, gdy znaczniki TCP w datagramie pasują do określonych przez `mask□` i `list□`. `maska` to lista rozdzielonych przecinkami znaczników, które powinny być sprawdzone przy przeprowadzaniu testu. `lista` to lista oddzielonych przecinkami, znaczników, które muszą być ustawione, aby reguła pasowała. Dopuszczalne znaczniki to *SYN*, *ACK*, *FIN*, *RST*, *URG*, *PSH*, *ALL* lub *NONE*. Jest to zaawansowana opcja. Zajrzyj do dobrego opisu protokołu TCP, na przykład do RFC-793, a znajdziesz tam opis znaczenia i działania każdego z tych znaczników. Znak `!` neguje regułę.

- *[!] --syn*

  Powoduje, że reguła pasuje tylko do datagramów z ustawionym bitem `SYN` i wyzerowanymi bitami `ACK` i `FIN`. Datagramy z tymi bitami są używane do otwierania połączeń TCP i dla tego ta opcja jest używana do obsługi żądań połączeń. Opcja ta to skrót od:

  ```
  - -tcp-flags SYN,RST,ACK SYN
  ```

  Gdy użyjesz operatora negacji, do reguły będą pasowały wszystkie datagramy, które nie mają ustawionych bitów `SYN` i `ACK`.

Rozszerzenia UDP: używane z –m udp –p udp

- *-sport[!][port[:port]]*

  Określa port, z którego musi pochodzić datagram, aby pasował do reguły. Porty mogą być podane jako zakres przez określenie górnego i dolnego limitu zakresu,

które należy rozdzielić dwukropkiem. Na przykład `20:25` oznacza wszystkie porty o numerach od 20 do 25 włącznie. Znów znak ! może być użyty do zanegowania wartości.

- *-dport[!][port[:port]]*
  Określa port, do którego musi być skierowany datagram, aby pasował do reguły. Argument jest kodowany tak samo jak *--sport*.

Rozszerzenia ICMP: używane z –m icmp –p icmp

- *--icmp-type[!] nazwa_typu*
  Określa typ komunikatu ICMP pasującego do reguły. Typ może być określony przez nazwę lub numer. Niektóre dopuszczalne nazwy to: `echo-request`, `echo-reply`, `source-quench`, `time-exceeded`, `destination-unreachable`, `network-unreachable`, `host-unreachable`, `protocol-unreachable` i `port-unreachable`.

Rozszerzenia MAC: używane z –m mac

- *--mac-source[!] adres*
  Określa adres hosta Ethernet, który wysłał datagram pasujący do tej reguły. Ma to sens jedynie w łańcuchach wejściowym i przekazującym reguły, ponieważ wszystkie datagramy, które przechodzą przez łańcuch wychodzący, są wysyłane przez nas.

Kolejna poprawiona wersja naszego prostego przykładu

Aby zaimplementować nasz prosty przykład za pomocą *netfilter*, mógłbyś załadować moduł *ipchains.o* i wykorzystać skrypt w wersji dla *ipchains*. Jednak zamiast tego, zaimplementowaliśmy go, używając *iptables*, by pokazać, jak bardzo te dwa programy są do siebie podobne.

Znów załóżmy, że mamy w firmie sieć i że używamy komputera z Liunksem jako firewalla, który pozwala naszym użytkownikom dostawać się do serwerów WWW w Internecie, ale nie przepuszcza innego ruchu.

Gdyby nasza sieć miała 24-bitową maskę (klasa C) i adres 172.16.1.0, użylibyśmy następujących reguł *iptables*:

```
# modprobe ip_tables
# iptables -F FORWARD
# iptables -P FORWARD DROP
# iptables -A FORWARD -m tcp -p tcp -s 0/0 --sport 80 -d 172.16.1.0/24 --syn -j DROP
# iptables -A FORWARD -m tcp -p tcp -s 172.16.1.0./24 --sport 80 -d 0/0 -j ACCEPT
# iptables -A FORWARD -m tcp -p tcp -d 172.16.1.0/24 --dport 80 -s 0/0 -j ACCEPT
```

W tym przykładzie polecenia *iptables* są interpretowane dokładnie tak jak ich równoważne polecenia *ipchains*. Główna różnica polega na tym, że musi być załadowany moduł *ip_tables.o*. Zauważ, że *iptables* nie obsługuje opcji *–b*, a więc musimy podać regułę dla każdego kierunku.

Operowanie bitem TOS

Bity typu usługi (*Type of Service* – TOS) to zestaw czterobitowych znaczników w nagłówku IP. Gdy dowolny z tych znaczników jest ustawiony, rutery mogą obsługiwać taki datagram inaczej niż datagramy bez ustawionych bitów TOS. Każdy z czterech bitów ma inne zadanie, a ustawiony może być tylko jeden z nich – kombinacja jest niedopuszczalna. Znaczniki są nazywane bitami typu usługi, ponieważ na kazują aplikacji wysyłającej dane informować sieć o typie wymaganej usługi sieciowej.

Dostępne klasy usług sieciowych:

Minimalne opóźnienie
: Używane, gdy czas potrzebny na przejście datagramu od hosta źródłowego do hosta docelowego (opóźnienie) jest bardzo ważny. Dostawca usług sieciowych może na przykład używać zarówno połączeń światłowodowych, jak i satelitarnych. Dane przesyłane przez satelitę pokonują dłuższą drogę, niż w przypadku sieci naziemnej pomiędzy tymi samymi punktami i ich opóźnienie jest dużo większe. Dostawca usług może spowodować, że datagramy z tym typem usługi na pewno nie będą przesyłane przez satelitę.

Maksymalna przepustowość
: Używana, gdy ważna jest liczba przesyłanych danych w dowolnym czasie. Istnieje wiele aplikacji sieciowych, dla których opóźnienie nie jest szczególnie ważne, ale przepustowość – tak. Na przykład masowe przesyłanie plików. Dostawca usług sieciowych może ustawić rutowanie datagramów tego typu usługi przez trasy o dużej przepustowości i dużych opóźnieniach, czyli np. połączenia satelitarne.

Maksymalna niezawodność
: Używana, gdy ważne jest, byś miał pewność, że dane dotrą do celu bez potrzeby ponownego przesyłania. Protokół IP może być przesyłany przez szereg różnych mediów transmisyjnych. Choć SLIP i PPP też się nadają do przenoszenia IP, nie są tak niezawodne jak sieć X.25. Dostawca usług sieciowych może udostępnić sieć alternatywną oferującą wysoką niezawodność i przeznaczoną do przesyłania IP, jeżeli zostanie wybrany ten typ usług.

Minimalny koszt
: Używany, gdy ważne jest zminimalizowanie kosztu przesyłania danych. Dzierżawienie przepustowości na satelicie dla połączeń przez ocean jest ogólnie tańsze, niż dzierżawienie kanału w światłowodzie na tej samej odległości, a więc dostawcy mogą udostępniać obie te możliwości, ale różnie liczyć koszty połączenia w zależności od tego, którego medium używasz. W tym scenariuszu usługa typu „minimalny koszt" może oznaczać, że datagramy będą kierowane przez tanie łącze satelitarne.

Ustawianie bitów TOS za pomocą ipfwadm i ipchains

Polecenia *ipfwadm* i *ipchains* obsługują bity TOS prawie tak samo. W obu przypadkach podajesz regułę, do której mają pasować datagramy z ustawionymi odpowiednimi bitami TOS, i używasz argumentu *–t* do określenia zmian, jakich chcesz dokonać.

Zmiany są określane za pomocą dwóch masek bitowych. Pierwsza z tych masek bitowych jest poddawana operacji logicznej AND z polem opcji IP datagramu, a druga jest poddawana logicznej operacji XOR z wynikiem poprzedniej operacji. Może wydawać się to skomplikowane, ale za chwilę wyjaśnimy, jak włącza się każdy z typów usług.

Maski bitowe są określane za pomocą ośmiobitowych wartości szesnastkowych. Zarówno *ipfwadm*, jak i *ipchains* używają tej samej składni przy zapisywaniu argumentów:

```
-t maskaand maskaxor
```

Na szczęście te same maski mogą być używane za każdym razem, gdy chcesz ustawić dany typ usługi, co zaoszczędzi ci każdorazowego ich rozpracowywania. W tabeli 9-3 pokazano je wraz z sugerowanym zastosowaniem.

Tabela 9-3. Sugerowane zastosowanie masek bitowych TOS

TOS	*Maska AND*	*Maska XOR*	*Sugerowane zastosowanie*
Minimalne opóźnienie	`0x01`	`0x10`	ftp, telnet, ssh
Maksymalna przepustowość	`0x01`	`0x08`	ftp-data, www
Maksymalna niezawodność	`0x01`	`0x04`	snmp, dns
Minimalny koszt	`0x01`	`0x02`	nntp, smtp

Ustawienie bitów TOS za pomocą iptables

Narzędzie *iptables* pozwala określić reguły, które przechwytują tylko datagramy z bitami TOS pasującymi do wartości określonej wcześniej, za pomocą opcji *–m tos*. Ustawienie bitów TOS datagramów IP pasujących do reguły następuje za pomocą celu `-j TOS`. Bity TOS możesz ustawiać tylko w łańcuchach `FORWARD` i `OUTPUT`. Dopasowanie i ustawienie są realizowane niezależnie od siebie. Możesz skonfigurować wszelkie reguły. Na przykład możesz skonfigurować regułę, która odrzuca wszystkie datagramy o pewnych kombinacjach bitów TOS, lub inną, która ustawia bity TOS datagramu pochodzącego tylko z pewnych hostów. Najczęściej będziesz używał reguł, które realizują zarówno dopasowanie, jak i zmianę, by za ich pomocą tłumaczyć TOS, podobnie jak możesz to zrobić w *ipfwadm* i *ipchains*.

Zamiast skomplikowanej, wykorzystującej dwie maski, konfiguracji stosowanej przez *ipfwadm* i *ipchains*, *iptables* proponuje prostsze rozwiązanie. Polega ono na jawnym określaniu bitów TOS, które powinny pasować, i tych, które powinny być ustawione. Co więcej, zamiast w wartościach szesnastkowych, możesz podać bity TOS za pomocą bardziej przyjaznych mnemonik, podanych w poniższej tabeli.

Ogólna składnia używana do dopasowania bitów TOS wygląda tak:

```
-m tos --tos mnemonik [inne-typy] -j cel
```

Ogólna składnia używana do ustawiania bitów TOS jest następująca:

```
[inne-argumenty] -j TOS --set mnemonik
```

Pamiętaj, że zwykle powinny być one używane razem, ale mogą być używane niezależnie, jeżeli twoja konfiguracja tego wymaga.

Mnemonika	*Wartość szesnastkowa*
Normal-Service	0x00
Minimize-Cost	0x02
Maximize-Reliability	0x04
Maximize-Throughput	0x08
Minimize-Delay	0x10

Testowanie konfiguracji firewalla

Gdy odpowiednio skonfigurowałeś firewall, trzeba sprawdzić, czy rzeczywiście działa tak, jak chcesz. Można w tym celu spróbować przebić się przez firewall testowym hostem spoza twojej sieci. Jest to sposób niezręczny, i wolny, a poza tym jest ograniczony do testowania jedynie tych adresów, których rzeczywiście używasz.

W implementacji firewalla w Linuksie dostępna jest szybsza i prostsza metoda. Pozwala ona na ręczne generowanie testów i uruchamianie ich z konfiguracją firewalla tak, jakbyś testował rzeczywiste datagramy. Wszystkie odmiany oprogramowania firewalla w Linuksie, *ipfwadm*, *ipchains* i *iptables*, udostępniają tego typu testowanie. Implementacja wykorzystuje polecenie *check*.

Ogólna procedura testowa wygląda następująco:

1. Zaprojektuj i skonfiguruj firewall, używając *ipfwadm*, *ipchains* lub *iptables*.
2. Przygotuj serię testów, które pokażą, czy twój firewall rzeczywiście działa tak, jak chciałeś. Do tych testów możesz użyć dowolnych adresów źródła i przeznaczenia, a więc wybierz jakąś mieszankę adresów, które będą mogły być przyjęte i które powinny być odrzucone. Jeżeli przepuszczasz i odrzucasz dane zakresy adresów, dobrze jest testować adresy z obu stron ograniczeń, tzn. jeden adres ze środka zakresu i jeden spoza niego. Pomoże to upewnić się, że masz skonfigurowane poprawne ograniczenia, ponieważ czasem zdarza się podać w konfiguracji niepoprawną maskę. Jeżeli filtrujesz według numerów protokołów i portów, twoje testy powinny również uwzględniać wszystkie istotne kombinacje tych parametrów. Na przykład, jeżeli zamierzasz przyjmować tylko pakiety TCP w pewnych warunkach, sprawdzaj, czy pakiety UDP są odrzucane.
3. Wykorzystaj reguły *ipfwadm*, *ipchains* lub *iptables* do implementacji każdego testu. Na pewno warto zapisać wszystkie reguły w skrypcie, tak byś mógł łatwo powtarzać testy, gdy zrobisz błąd lub zmienisz budowę. Testy wykorzystują prawie tę samą składnię co reguły, ale argumenty mają nieco inne znaczenie. Na przykład argument adresu źródłowego w regule określa adres źródłowy, który powinien mieć datagram, by pasować do danej reguły. Adres źródłowy w składni testowej

dla odmiany oznacza adres źródłowy datagramu testowego, który zostanie wygenerowany. W przypadku *ipfwadm* musisz użyć opcji *–c*, by określić, że to polecenie jest testowe, natomiast w *ipchains* i *iptables* robisz to za pomocą opcji *–C*. We wszystkich przypadkach musisz *zawsze* podawać adres źródłowy, adres docelowy, protokół i interfejs, które mają być użyte w teście. Inne argumenty, takie jak numery portów czy ustawienie bitów TOS, są opcjonalne.

4. Wykonaj każde polecenie testowe i zapisz wynik. Wynik każdego testu będzie miał postać jednego słowa wskazującego ostateczne przeznaczenie datagramu po przejściu przez konfigurację firewalla – to znaczy po zakończeniu przetwarzania. W przypadku *ipchains* i *iptables*, poza łańcuchami wbudowanymi, będą testowane łańcuchy definiowane przez użytkownika.
5. Porównaj wynik każdego testu z oczekiwanym rezultatem. Jeżeli widzisz jakieś różnice, musisz przeanalizować swój zestaw reguł, by stwierdzić, gdzie zrobiłeś błąd. Jeżeli zapisałeś polecenia testowe w skrypcie, możesz łatwo powtórzyć test po poprawieniu wszelkich błędów w konfiguracji firewalla. Dobrze jest zupełnie skasować zestaw reguł i stworzyć je od nowa, a nie dokonywać zmiany dynamicznie. Pomaga to upewnić się, że aktywna konfiguracja, którą testujesz, rzeczywiście odzwierciedla zestaw poleceń w twoim skrypcie konfiguracyjnym.

Przyjrzyjmy się, jak może wyglądać zapis ręcznego testowania naszego przykładu w przypadku *ipchains*. Pamiętasz, że nasza przykładowa sieć lokalna ma adres 172.16.1.0 i maskę sieciową 255.255.255.0, i że pozwoliliśmy na realizowanie połączeń do serwerów WWW w sieci. Nic więcej nie powinno przechodzić przez nasz łańcuch przekazywania. Rozpocznijmy od testowania tego, o czym wiemy, że powinno działać – połączenia z lokalnego hosta do serwera WWW na zewnątrz:

```
# ipchains -C forward -p tcp -s 172.16.1.0 1025 -d 44.136.8.2 80 -i eth0
accepted
```

Zwróć uwagę, które argumenty muszą być podane i w jaki sposób zostały użyte do opisania datagramu. Wynik polecenia wskazuje, że datagram został przyjęty do przekazania, czyli jest zgodny z naszymi oczekiwaniami.

Teraz spróbujmy innego testu; tym razem adres źródłowy nie należy do naszej sieci. Pakiet nie powinien przejść:

```
# ipchains -C forward -p tcp -s 172.16.2.0 1025 -d 44.136.8.2 80 -i eth0
denied
```

Zróbmy kilka innych testów, tym razem z kilkoma szczegółami identycznymi z pierwszym testem, ale innymi protokołami. Te pakiety też nie powinny zostać przepuszczone.

```
# ipchains -C forward -p udp -s 172.16.1.0 1025 -d 44.136.8.2 80 -i eth0
denied
# ipchains -C forward -p icmp -s 172.16.1.0 1025 -d 44.136.8.2 80 -i eth0
denied
```

Sprawdźmy inny port docelowy. Znów oczekujemy, że pakiet zostanie odrzucony:

```
# ipchains -C forward -p tcp -s 172.16.1.0 1025 -d 44.136.8.2 23 -i eth0
denied
```

Musisz przebyć długą drogę, zanim uspokoisz swoje myśli, wykonując szereg wyczerpujących testów. Choć czasem może to być równie trudne jak skonfigurowanie firewalla, jest to najlepszy sposób, by wiedzieć, że twój projekt zapewnia najlepsze bezpieczeństwo, jakiego możesz oczekiwać.

Przykładowa konfiguracja firewalla

Omówiliśmy podstawowe konfiguracje firewalla. Przyjrzyjmy się, jak w praktyce taka konfiguracja może wyglądać.

Konfiguracja pokazana w tym przykładzie została zaprojektowana tak, by było ją łatwo rozbudować i dostosować do własnych potrzeb. Pokazaliśmy trzy wersje. Pierwsza jest zaimplementowana za pomocą polecenia *ipfwadm* (lub skryptu *ipfwadm-wrapper*), druga wykorzystuje *ipchains*, a trzecia *iptables*. Przykłady nie wykorzystują łańcuchów definiowanych przez użytkownika, ale pokazują podobieństwa i różnice pomiędzy starymi i nowymi składniami narzędzi do konfiguracji firewalla:

```
#!/bin/bash
##############################################################
# WERSJA IPFWADM
# Ta przyk□adowa konfiguracja jest przeznaczona dla jednego
# hosta i nie uwzgl□dnia us□ug oferowanych przez sam komputer,
# na którym dzia□a firewall.
##############################################################

# CZ□□□ DO KONFIGURACJI PRZEZ U□YTKOWNIKA

# Nazwa i lokalizacja programu ipfwadm. U□yj ipfwadm-wrapper
# w przypadku j□der 2.2.*
IPFWADM=ipfwadm

# □cie□ka do pliku wykonywalnego ipfwadm
PATH="/sbin"

# Przestrze□ adresowa naszej sieci wewn□trznej i urz□dzenie j□ obs□uguj□ce
OURNET="172.29.16.0/24"
OURBCAST="172.29.16.255"
OURDEV="eth0"

# Adres zewn□trzny i urz□dzenie sieciowe go obs□uguj□ce
ANYADDR="0/0"
ANYDEV="eth1"

# Us□ugi TCP, które chcemy przepuszcza□ - "" puste oznacza wszystkie porty
# uwaga: oddzielone spacj□
TCPIN="smtp www"
TCPOUT="smtp www ftp ftp-data irc"

# Us□ugi UDP, które chcemy przepuszcza□ - "" puste oznacza wszystkie porty
# uwaga: oddzielone spacj□
UDPIN="domain"
UDPOUT="domain"
```

```
# Us□ugi ICMP, które chcemy przepuszcza□ - "" puste oznacza wszystkie typy
# numery typów us□ug znajdziesz w pliku /usr/include/netinet/ip_icmp.h
# uwaga: oddzielone spacj□
ICMPIN="0 3 11"
ICMPOUT="8 3 11"

# Logowanie; usu□ komentarz z poni□szego wiersza, by w□□czy□
# zapisywanie datagramów, które nie s□ przepuszczane przez
# firewall
# LOGGING=1

# KONIEC CZ□□CI KONFIGUROWALNEJ PRZEZ U□YTKOWNIKA
##############################################################
# Usuni□cie tablicy regu□ przychodz□cych
$IPFWADM -I -f

# Chcemy domy□lnie odrzuca□ ruch przychodz□cy
$IPFWADM -I -p deny

# PODSZYWANIE SI□ (SPOOFING)
# Nie powinni□my przyjmowa□ datagramów, które maj□ adres
# □ród□owy z naszej sieci, ale pakiet przychodzi z
# zewn□trz, a wi□c go odrzucamy
$IPFWADM -I -a deny -S $OURNET -W $ANYDEV

# SMURF
# Zabraniamy wys□ania pakietów ICMP na nasz adres
# rozg□oszeniowy, by zapobiec atakowi typu "Smurf"
$IPFWADM -I -a deny -p icmp -W $ANYDEV -D $OURBCAST

# TCP
# B□dziemy przyjmowali wszystkie datagramy TCP nale□□ce do
# istniej□cego po□□czenia (tj. posiadaj□ce ustawiony bit ACK)
# z portem TCP, który przepuszczamy. Powinno to obj□□
# ponad 95 % wszystkich poprawnych pakietów TCP.
$IPFWADM -I -a accept -P tcp -D $OURNET $TCPIN -k -b

# TCP - PO□□CZENIA PRZYCHODZ□CE
# B□dziemy przyjmowali □□dania po□□cze□ z zewn□trz tylko na
# dozwolone porty TCP
$IPFWADM -I -a accept -P tcp -W $ANYDEV -D $OURNET $TCPIN -y

# TCP - PO□□CZENIA WYCHODZ□CE
# Przyjmujemy wszystkie □□dania wychodz□cych po□□cze□ tcp na
# dozwolone porty TCP.
$IPFWADM -I -a accept -P tcp -W $OURDEV -D $ANYADDR $TCPOUT -y

# UDP - PRZYCHODZ□CE
# Przepuszczamy wszystkie datagramy UDP przychodz□ce na
# dozwolone porty
$IPFWADM -I -a accept -P udp -W $ANYDEV -D $OURNET $UDPIN

# UDP - WYCHODZ□CE
# Przepuszczamy wszystkie datagramy UDP wychodz□ce na
# dozwolone porty
$IPFWADM -I -a accept -P udp -W $OURDEV -D $ANYADDR $UDPOUT

# ICMP - PRZYCHODZ□CE
# Przepuszczamy wszystkie przychodz□ce datagramy ICMP o
# dopuszczalnych typach
```

```
$IPFWADM -I -a accept -P icmp -W $ANYDEV -D $OURNET $ICMPIN

# ICMP - WYCHODZ□CE
# Przepuszczamy wszystkie wychodz□ce datagramy ICMP o
# dopuszczalnych typach
$IPFWADM -I -a accept -P icmp -W $OURDEV -D $ANYADDR $ICMPOUT

# DEFAULT i LOGGING
# Wszystkie pozosta□e datagramy trafiaj□ do regu□y domy□lnej i
# s□ odrzucane. Je□eli skonfigurujesz wcze□niej zmienn□
# LOGGING, b□d□ one zapisywane.
#
if [ "$LOGGING" ]
then
   # Zapisywanie odrzuconych pakietów TCP
   $IPFWADM -I -a reject -P tcp -o

   # Zapisywanie odrzuconych pakietów UDP
   $IPFWADM -I -a reject -P udp -o

   # Zapisywanie odrzuconych pakietów ICMP
   $IPFWADM -I -a reject -P icmp -o
fi
#
#end.
```

Teraz zaimplementujemy to samo za pomocą polecenia *ipchains*:

```
#!/bin/bash
##############################################################
# WERSJA IPCHAINS
# Ta przyk□adowa konfiguracja jest przeznaczona dla jednego
# hosta i nie uwzgl□dnia us□ug oferowanych przez sam komputer,
# na którym dzia□a firewall.
##############################################################

# CZ□□□ KONFIGUROWALNA PRZEZ U□YTKOWNIKA

# Nazwa i lokalizacja programu ipchains.
IPCHAINS=ipchains

# □cie□ka do pliku wykonywalnego ipchains
PATH="/sbin"

# Przestrze□ adresowa naszej sieci wewn□trznej i urz□dzenie j□ obs□uguj□ce
OURNET="172.29.16.0/24"
OURBCAST="172.29.16.255"
OURDEV="eth0"

# Adres zewn□trzny i urz□dzenie sieciowe go obs□uguj□ce
ANYADDR="0/0"
ANYDEV="eth1"

# Us□ugi TCP, które chcemy przepuszcza□ - "" puste oznacza wszystkie porty
# uwaga: oddzielone spacj□
TCPIN="smtp www"
TCPOUT="smtp www ftp ftp-data irc"

# Us□ugi UDP, które chcemy przepuszcza□ - "" puste oznacza wszystkie porty
```

```
# uwaga: oddzielone spacj□
UDPIN="domain"
UDPOUT="domain"

# Us□ugi ICMP, które chcemy przepuszcza□ - "" puste oznacza wszystkie typy
# numery typów us□ug znajdziesz w pliku
# /usr/include/netinet/ip_icmp.h
# uwaga: oddzielone spacj□
ICMPIN="0 3 11"
ICMPOUT="8 3 11"

# Logowanie; usu□ komentarz z poni□szego wiersza, by w□□czy□
# zapisywanie datagramów, które nie s□ przepuszczane przez
# firewall
# LOGGING=1

# KONIEC CZ□□CI KONFIGUROWALNEJ PRZEZ U□YTKOWNIKA
###########################################################
# Usuni□cie tablicy regu□ przychodz□cych
$IPCHAINS -F input

# Chcemy domy□lnie odrzuca□ ruch przychodz□cy
$IPCHAINS -P input deny

# PODSZYWANIE SI□ (SPOOFING)
# Nie powinni□my przyjmowa□ datagramów, w których adres
# □ród□owy jest z naszej sieci, ale pakiet przychodzi z
# zewn□trz, a wi□c go odrzucamy
$IPCHAINS -A input -s $OURNET -i $ANYDEV -j deny

# SMURF
# Zabraniamy wys□ania pakietów ICMP na nasz adres
# rozg□oszeniowy, by zapobiec atakowi typu "Smurf"
$IPCHAINS -A input -p icmp -w $ANYDEV -d $OURBCAST -j deny

# Powinni□my przyjmowa□ fragmenty, w ipchains musimy to
# zadeklarowa□ jawnie
$IPCHAINS -A input -f -j accept

# TCP
# B□dziemy przyjmowali wszystkie datagramy TCP nale□□ce do
# istniej□cego po□□czenia (tj. posiadaj□ce ustawiony bit ACK)
# z portem TCP, który przepuszczamy. Powinno to obj□□
# ponad 95 % wszystkich poprawnych pakietów TCP.
$IPCHAINS -A input -p tcp -d $OURNET $TCPIN ! -y -b -j accept

# TCP - PO□□CZENIA PRZYCHODZ□CE
# B□dziemy przyjmowali □□dania po□□cze□ z zewn□trz tylko na
# dozwolone porty TCP
$IPCHAINS -A input -p tcp -i $ANYDEV -d $OURNET $TCPIN -y -j accept

# TCP - PO□□CZENIA WYCHODZ□CE
# Przyjmujemy wszystkie □□dania wychodz□cych po□□cze□ tcp na
# dozwolone porty TCP.
$IPCHAINS -A input -p tcp -i $OURDEV -d $ANYADDR $TCPOUT -y -j accept

# UDP - PRZYCHODZ□CE
# Przepuszczamy wszystkie datagramy UDP przychodz□ce na
# dozwolone porty
$IPCHAINS -A input -p udp -i $ANYDEV -d $OURNET $UDPIN -j accept
```

```
# UDP - WYCHODZ□CE
# Przepuszczamy wszystkie datagramy UDP wychodz□ce na
# dozwolone porty
$IPCHAINS -A input -p udp -i $OURDEV -d $ANYADDR $UDPOUT -j accept

# ICMP - PRZYCHODZ□CE
# Przepuszczamy wszystkie przychodz□ce datagramy ICMP o
# dopuszczalnych typach
$IPCHAINS -A input -p icmp -w $ANYDEV -d $OURNET $ICMPIN -j accept

# ICMP - WYCHODZ□CE
# Przepuszczamy wszystkie wychodz□ce datagramy ICMP o
# dopuszczalnych typach
$IPCHAINS -A input -p icmp -i $OURDEV -d $ANYADDR $ICMPOUT -j accept

# DEFAULT i LOGGING
# Wszystkie pozosta□e datagramy trafiaj□ do regu□y domy□lnej i
# s□ odrzucane. Je□eli skonfigurujesz wcze□niej zmienn□
# LOGGING, b□d□ one zapisywane.
#
if [ "$LOGGING" ]
then\


   # Zapisywanie odrzuconych pakietów TCP
   $IPCHAINS -A input -p tcp -l -j reject

   # Zapisywanie odrzuconych pakietów UDP
   $IPCHAINS -A input -p udp -l -j reject

   # Zapisywanie odrzuconych pakietów ICMP
   $IPCHAINS -A input -p icmp -l -j reject
fi
#
#end.
```

W naszym przykładzie *iptables* przeszliśmy na korzystanie z reguły `FORWARD`, ze względu na różnicę w znaczeniu zestawu reguł `INPUT` w implementacji *netfilter*. Jest to dla nas istotne; oznacza, że żadna z reguł nie broni samego hosta firewalla. Aby dokładnie naśladować przykład *ipchains*, skopiujemy każdą z naszych reguł do łańcucha `INPUT`. Dla jasności odrzuciliśmy wszystkie przychodzące datagramy odebrane po zewnętrznej stronie naszego interfejsu.

```
#!/bin/bash
##############################################################
# WERSJA IPTABLES
# Ta przyk□adowa konfiguracja jest przeznaczona dla jednego
# hosta i nie uwzgl□dnia us□ug oferowanych przez sam komputer,
# na którym dzia□a firewall.
##############################################################

# CZ□□□ KONFIGUROWALNA PRZEZ U□YTKOWNIKA

# Nazwa i lokalizacja programu iptables.
IPTABLES=iptables

# □cie□ka do pliku wykonywalnego ipchains
PATH="/sbin"
```

```
# Przestrze□ adresowa naszej sieci wewn□trznej i urz□dzenie j□
# obs□uguj□ce
OURNET="172.29.16.0/24"
OURBCAST="172.29.16.255"
OURDEV="eth0"

# Adres zewn□trzny i urz□dzenie sieciowe go obs□uguj□ce
ANYADDR="0/0"
ANYDEV="eth1"

# Us□ugi TCP, które chcemy przepuszcza□ - "" puste oznacza wszystkie porty
# uwaga: oddzielone przecinkami
TCPIN="smtp,www"
TCPOUT="smtp,www,ftp,ftp-data,irc"

# Us□ugi UDP, które chcemy przepuszcza□ - "" puste oznacza wszystkie porty
# uwaga: oddzielone przecinkami
UDPIN="domain"
UDPOUT="domain"

# Us□ugi ICMP, które chcemy przepuszcza□ - "" puste oznacza wszystkie typy
# numery typów us□ug znajdziesz w pliku
# /usr/include/netinet/ip_icmp.h
# uwaga: oddzielone przecinkami
ICMPIN="0,3,11"
ICMPOUT="8,3,11"

# Logowanie; usu□ komentarz z poni□szego wiersza, by w□□czy□
# zapisywanie datagramów, które nie s□ przepuszczane przez
# firewall
# LOGGING=1

# KONIEC CZ□□CI KONFIGUROWALNEJ PRZEZ U□YTKOWNIKA
###########################################################
# Usuni□cie tablicy regu□ przychodz□cych
$IPTABLES -F FORWARD

# Chcemy domy□lnie odrzuca□ ruch przychodz□cy
$IPTABLES -P FORWARD deny

# Odrzucamy wszystkie datagramy pochodz□ce z zewn□trz i
# przeznaczone dla tego hosta
$IPTABLES -A INPUT -i $ANYDEV -j DROP

# PODSZYWANIE SI□ (SPOOFING)
# Nie powinni□my przyjmowa□ datagramów, w których adres
# □ród□owy jest z naszej sieci, ale pakiet przychodzi z
# zewn□trz, a wi□c go odrzucamy
$IPTABLES -A FORWARD -s $OURNET -i $ANYDEV -j DROP

# SMURF
# Zabraniamy wys□ania pakietów ICMP na nasz adres
# rozg□oszeniowy, by zapobiec atakowi typu "Smurf"
$IPTABLES -A FORWARD -m multiport -p icmp -i $ANYDEV -d $OURBCAST -j DENY

# Powinni□my przyjmowa□ fragmenty, w iptables musimy to
# zadeklarowa□ jawnie
$IPTABLES -A FORWARD -f -j ACCEPT

# TCP
```

```
# B□dziemy przyjmowali wszystkie datagramy TCP nale□□ce do
# istniej□cego po□□czenia (tj. posiadaj□ce ustawiony bit ACK)
# z portem TCP, który przepuszczamy. Powinno to obj□□
# ponad 95 % wszystkich poprawnych pakietów TCP.
$IPTABLES -A FORWARD -m multiport -p tcp -d $OURNET --dports $TCPIN /
    ! --tcp-flags SYN,ACK ACK -j ACCEPT
$IPTABLES -A FORWARD -m multiport -p tcp -s $OURNET --sports $TCPIN /
    ! --tcp-flags SYN,ACK ACK -j ACCEPT

# TCP - PO□□CZENIA PRZYCHODZ□CE
# B□dziemy przyjmowali □□dania po□□cze□ z zewn□trz tylko na
# dozwolone porty TCP
$IPTABLES -A FORWARD -m multiport -p tcp -i $ANYDEV -d $OURNET $TCPIN /
    --syn -j ACCEPT

# TCP - PO□□CZENIA WYCHODZ□CE
# Przyjmujemy wszystkie □□dania wychodz□cych po□□cze□ tcp na
# dozwolone porty TCP.
$IPTABLES -A FORWARD -m multiport -p tcp -i $OURDEV -d $ANYADDR /
    --dport $TCPOUT --syn -j ACCEPT

# UDP - PRZYCHODZ□CE
# Przepuszczamy wszystkie datagramy UDP przychodz□ce na
# dozwolone porty
$IPTABLES -A FORWARD -m multiport -p udp -i $ANYDEV -d $OURNET /
    --dports $UDPIN -j ACCEPT
$IPTABLES -A FORWARD -m multiport -p udp -i $ANYDEV -s $OURNET /
    --sports $UDPIN -j ACCEPT

# UDP - WYCHODZ□CE
# Przepuszczamy wszystkie datagramy UDP wychodz□ce na
# dozwolone porty
$IPTABLES -A FORWARD -m multiport -p udp -i $OURDEV -d $ANYADDR /
    --dports $UDPOUT -j ACCEPT
$IPTABLES -A FORWARD -m multiport -p udp -i $OURDEV -s $ANYADDR /
    --sports $UDPOUT -j ACCEPT

# ICMP - PRZYCHODZ□CE
# Przepuszczamy wszystkie przychodz□ce datagramy ICMP o
# dopuszczalnych typach
$IPTABLES -A FORWARD -m multiport -p icmp -i $ANYDEV -d $OURNET /
    --dports $ICMPIN -j ACCEPT

# ICMP - WYCHODZ□CE
# Przepuszczamy wszystkie wychodz□ce datagramy ICMP o
# dopuszczalnych typach
$IPTABLES -A FORWARD -m multiport -p icmp -i $OURDEV -d $ANYADDR /
    --dports $ICMPOUT -j ACCEPT

# DEFAULT i LOGGING
# Wszystkie pozosta□e datagramy trafiaj□ do regu□y domy□lnej i
# s□ odrzucane. Je□eli skonfigurujesz wcze□niej zmienn□
# LOGGING, b□d□ one zapisywane.
#
if [ "$LOGGING" ]
then
   # Zapisywanie odrzuconych pakietów TCP
   $IPTABLES -A FORWARD -m tcp -p tcp -l -j LOG
```

```
   # Zapisywanie odrzuconych pakietów UDP
   $IPTABLES -A FORWARD -m udp -p udp -l -j LOG

   # Zapisywanie odrzuconych pakietów ICMP
   $IPTABLES -A FORWARD -m icmp -p icmp -l -j LOG
fi
#
#end.
```

W wielu prostych sytuacjach, aby skorzystać z pokazanych przykładów, wystarczy dokonać edycji początkowej części skryptu zatytułowanej „CZĘŚĆ DO KONFIGURACJI PRZEZ UŻYTKOWNIKA”, aby określić, które protokoły i typy datagramów chcesz przepuszczać w obie strony. Przy bardziej złożonych konfiguracjach będziesz musiał dokonać także edycji pozostałej części skryptu. Pamiętaj – jest to prosty przykład, a więc podczas jego implementacji przeanalizuj go bardzo uważnie, by robił to, co chcesz.

10

Liczenie ruchu IP



W świecie komercyjnych usług internetowych coraz ważniejsza staje się wiedza o tym, ile da nych wy syłasz i ile od bier asz przez swo je połącze nia sie ciowe. Przy da ci się, je żeli je steś do stawcą usług in tern eto wych i wy staw iasz ra chunki swo im klientom według iloś ci przesłanych da nych. Na to miast jeżeli je steś klien tem do stawcy usług, który ob ciąża cię według iloś ci przesłanych da nych, war to sa me mu zbie rać da ne, aby spraw dzać ra chunki wy staw iane przez do stawcę.

Są jesz cze in ne za stos owa nia li czen ia ru chu, nie mające nic wspólnego z pie niędz mi ani ra chunk ami. Je żeli zarządzasz ser wer em, któ ry udos tępnia różne ty py usług sieciow ych, przy datna może być wie dza o tym, ile da nych ge ner uje ka żda z nich. Te go ty pu in form acje mogą ci pomóc przy po dejm owa niu de cyz ji o tym, ja ki sprzęt ku pić lub ile serwerów uruc homić.

Jądro Linuk sa udos tępnia funk cję po zwal ającą zbie rać wszel kiego ro dzaju przy datne in form acje o ru chu sie ciow ym. Funk cja ta to *li czen ie ru chu IP* (ang. *IP acc oun ting*).

Konfigurowanie jądra do liczenia ruchu IP

Funk cja li czen ia ru chu IP w Linuksie jest bli sko związa na z oprog ramo wan iem firewalla. Miejsca, z których chcesz zbie rać da ne roz lic zeni owe, to te sa me miej sca, które in ter esują cię przy fil trow aniu przez fi rew all: we jśc ie i wy jśc ie z ho sta do sie c i oraz oprog ramo wan ie ru tujące da tag ramy. Je żeli jesz cze nie prze czytałeś roz działu o fi rew alla ch, to te raz jest do skonały mo ment, żeby to zro bić oraz wy kor zyst ać kil ka po jęć opis any ch w roz dziale 9, *Firewall TCP/IP*.

Aby włączyć funk cję li cze nia ru chu IP w Linuksie, po wi nie neś naj pierw zo ba czyć, czy jądro jest odpowiednio skonfigurowane. Sprawdź, czy istnieje plik */proc/net/ip_acct*. Je żeli ist nie je, two je jądro już obsługu je li cze nie ru chu IP. Je żeli nie ist nie je, mu sisz skom pi lo wać jądro od no wa, pil nując, byś od po wie dział „Y” na po niż sze pyta nia w jądrach se rii 2.0 i 2.2.

```
Networking options --->
       [*] Network firewalls
       [*] TCP/IP networking
        ...
       [*] IP: accounting
```

lub w jądrach serii 2.4:

```
Networking options --->
    [*] Network packet filtering (replaces ipchains)
```

Konfigurowanie liczenia ruchu IP

Ponieważ usługa liczenia ruchu IP jest ściśle związana z firewallem IP, do jej konfigurowania służą te same narzędzia, czyli *ipfwadm*, *ipchains* lub *iptables*. Składnia polecenia jest bardzo podobna jak w przypadku reguł firewalla, a więc nie będziemy się na niej skupiać, a omówimy to, czego możesz się dowiedzieć o swojej sieci, używając tej funkcji.

Ogólna składnia polecenia liczącego ruch IP dla *ipfwadm* jest następująca:

```
# ipfwadm -A [kierunek] [polecenie] [parametry]
```

Nowy jest argument kierunku. Przyjmuje on jedną z wartości `in`, `out` lub `both`. Są to kierunki ruchu z punktu widzenia samego komputera z Linuksem, a więc `in` oznacza dane przychodzące z sieci do komputera, a `out` oznacza dane wysyłane przez hosta do sieci. Kierunek `both` stanowi sumę danych przychodzących i wychodzących.

Ogólna składnia polecenia dla *ipchains* i *iptables* jest następująca:

```
# ipchains -A łańcuch definicja-reguły
# iptables -A łańcuch definicja-reguły
```

Polecenia *ipchains* i *iptables* pozwalają określić kierunek w sposób bardziej spójny z regułami firewalla. Łańcuchy IP nie pozwalają na konfigurowanie reguł, które obejmują oba kierunki, ale dopuszczają skonfigurowanie reguł w łańcuchu `forward`, co nie było możliwe w starszej implementacji. W kilku przykładach pokazanych dalej zobaczymy, co z tego wyniknie.

Polecenia są w dużym stopniu takie same jak w regułach firewalla, z tą różnicą, że nie używa się tu polityk. Możemy dodawać, wstawiać, usuwać i listować reguły liczenia ruchu. W przypadku *ipchains* i *iptables* dopuszczalne są tylko reguły liczenia ruchu, a wszelkie polecenia niezawierające opcji *–j* realizują jedynie liczenie ruchu.

Parametry w definicji reguły liczenia ruchu IP są identyczne z używanymi dla firewalla IP. Za ich pomocą definiujemy dokładnie, jaki ruch sieciowy chcemy liczyć.

Liczenie według adresu

Na przykładzie pokażemy, jak korzysta się z funkcji liczenia ruchu IP.

Wyobraź sobie, że mamy ruter oparty na Linuksie, który obsługuje dwa wydziały browaru wirtualnego. Ruter ma dwa urządzenia Ethernet, *eth0* i *eth1*, z których każde obsługuje jeden wydział, oraz urządzenie PPP, *ppp0*, które łączy nas za pomocą szybkiego łącza szeregowego z głównym campusem uniwersytetu Groucho Marx.

Wyobraźmy sobie również, że dla celów rozliczeniowych chcemy znać całkowity ruch generowany przez każdy wydział podłączony przez łącze szeregowe, a dla celów zarządzania chcemy znać całkowity ruch generowany pomiędzy wydziałami.

Poniższa tabela pokazuje adresy interfejsów, których będziemy używać w naszym przykładzie:

iface	*adres*	*maska sieci*
eth0	172.16.3.0	255.255.255.0
eth1	172.16.4.0	255.255.255.0

Aby odpowiedzieć na pytanie: „Jak dużo danych na łączu PPP generuje każdy wydział?”, powinniśmy użyć następującego zestawu reguł:

```
# ipfwadm -A both -a W ppp0 -S 172.16.3.0/24 -b
# ipfwadm -A both -a W ppp0 -S 172.16.4.0/24 -b
```

lub

```
# ipchains -A input -i ppp0 -d 172.16.3.0/24
# ipchains -A output -i ppp0 -s 172.16.3.0/24
# ipchains -A input -i ppp0 -d 172.16.4.0/24
# ipchains -A output -i ppp0 -s 172.16.4.0/24
```

i w przypadku *iptables*:

```
# iptables -A FORWARD -i ppp0 -d 172.16.3.0/24
# iptables -A FORWARD -o ppp0 -s 172.16.3.0/24
# iptables -A FORWARD -i ppp0 -d 172.16.4.0/24
# iptables -A FORWARD -o ppp0 -s 172.16.4.0/24
```

Pierwsza połowa każdego z tych zestawów mówi: „Licz wszystkie dane przechodzące w obu kierunkach przez interfejs o nazwie `ppp0` z adresami źródłowym lub docelowym (pamiętaj o funkcji *–b* w przypadku *ipfwadm* i *ipchains*) `172.16.3.0/24`”. Druga połowa każdego zestawu reguł jest taka sama, ale dotyczy drugiej sieci Ethernet.

Aby odpowiedzieć na drugie pytanie: „Ile danych jest przesyłanych pomiędzy dwoma wydziałami?”, potrzebujemy następujących reguł:

```
# ipfwadm -A both -a -S 172.16.3.0/24 -D 172.16.4.0/24 -b
```

lub:

```
# ipchains -A forward -s 172.16.3.0/24 -d 172.16.4.0/24 -b
```

lub:

```
# iptables -A FORWARD -s 172.16.3.0/24 -d 172.16.4.0/24
# iptables -A FORWARD -s 172.16.4.0/24 -d 172.16.3.0/24
```

Te zestawy reguł liczą wszystkie datagramy, których adres źródłowy należy do sieci jednego wydziału, a adres docelowy – do sieci drugiego wydziału.

Liczenie ruchu według portu usługi

W porządku, załóżmy teraz, że chcemy wiedzieć coś o tym, jaki rodzaj ruchu jest przesyłany przez nasze łącze PPP. Możemy na przykład dowiedzieć się, jaką część łącza zajmują usługi FTP, smtp i WWW.

Skrypt z regułami włączający mi zbieranie tego typu informacji może wyglądać następująco:

```
#!/bin/sh
# Zbieranie, za pomoc□ ipfwadm, statystyk o ruchu FTP, smtp i www
#dla danych przesy□anych przez □□cze PPP
#
ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 ftp ftp-data
ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 smtp
ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 www
```

lub:

```
#!/bin/sh
# Zbieranie, za pomoc□ ipchains, statystyk o ruchu FTP, smtp i www
# dla danych przesy□anych przez □□cze PPP
#
ipchains -A input -i ppp0 -p tcp -s 0/0 ftp-data:ftp
ipchains -A output -i ppp0 -p tcp -d 0/0 ftp-data:ftp
ipchains -A input -i ppp0 -p tcp -s 0/0 smtp
ipchains -A output -i ppp0 -p tcp -d 0/0 smtp
ipchains -A input -i ppp0 -p tcp -s 0/0 www
ipchains -A output -i ppp0 -p tcp -d 0/0 www
```

lub:

```
#!/bin/sh
# Zbieranie, za pomoc□ iptables, statystyk o ruchu FTP, smtp i www
# dla danych przesy□anych przez □□cze PPP
#
iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport ftp-data:ftp
iptables -A FORWARD -o ppp0 -m tcp -p tcp --dport ftp-data:ftp
iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport smtp
iptables -A FORWARD -o ppp0 -m tcp -p tcp --dport smtp
iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport www
iptables -A FORWARD -o ppp0 -m tcp -p tcp --dport www
```

W tej konfiguracji jest kilka ciekawostek. Po pierwsze, określiliśmy protokół. Gdy w regułach podajemy numery portów, musimy także podać protokół, ponieważ TCP i UDP posiadają oddzielne zestawy portów. Ponieważ wszystkie z tych usług są oparte na TCP, podaliśmy go jako protokół. Po drugie, podaliśmy dwie usługi `ftp` i `ftp-data` w jednym poleceniu. *ipfwadm* pozwala na podawanie poszczególnych portów, zakresów portów lub losowych list portów. Polecenie *ipchains* również pozwala na podawanie poszczególnych portów lub zakresów portów, z czego skorzystaliśmy. Składnia „`ftp-data:ftp`" oznacza „porty od ftp-data (20) do ftp (21)" i jest to sposób kodowania zakresu portów w poleceniach *ipchains* i *iptables*. Gdy w regule zliczającej podasz listę portów, oznacza to, że wszelkie dane odebrane na wskazanych portach będą sumowane przy zliczaniu. Pamiętając, że FTP używa dwóch portów, portu poleceń i danych, podaliśmy je razem, żeby sumować cały ruch FTP. Na końcu podaliśmy adres źródłowy w postaci „`0/0`", co jest szczegól-

nym zapisem, do którego pasują wszystkie adresy; taki zapis jest wymagany przez polecenia *ipfwadm* i *ipchains*, aby można było określić porty.

Możemy się nieco bardziej skupić na drugim punkcie, co da nam inne spojrzenie na dane na naszym łączu. Wyobraźmy sobie, że traktujemy ruch FTP, SMTP i WWW jako istotny, a pozostały ruch jako nieistotny. Gdybyśmy chcieli znać stosunek ruchu istotnego do nieistotnego, moglibyśmy użyć czegoś takiego:

```
# ipfwadm -A both -a W ppp0 -P tcp -S 0/0 ftp ftp-data smtp www
# ipfwadm -A both -a W ppp0 -P tcp -S 0/0 1:19 22:24 26:79 81:32767
```

Jeżeli już przejrzałeś swój plik */etc/services*, wiesz, że druga reguła obejmuje wszystkie porty poza wymienionymi w pierwszej (`ftp`, `ftp-data`, `smtp` i `www`).

Jak to robimy w poleceniach *ipchains* i *iptables*, skoro pozwalają one określić tylko jeden port jako argument? Do liczenia ruchu możemy równie łatwo jak w regułach firewalla wykorzystać łańcuchy definiowane przez użytkownika. Rozważmy następujące podejście:

```
# ipchains -N a-essent
# ipchains -N a-noness
# ipchains -A a-essent -j ACCEPT
# ipchains -A a-noness -j ACCEPT
# ipchains -A forward -i ppp0 -p tcp -s 0/0 ftp-data:ftp -j a-essent
# ipchains -A forward -i ppp0 -p tcp -s 0/0 smtp -j a-essent
# ipchains -A forward -i ppp0 -p tcp -s 0/0 www -j a-essent
# ipchains -A forward -j a-noness
```

Tworzymy tutaj dwa łańcuchy definiowane przez użytkownika, jeden o nazwie `a-essent`, w którym zbieramy dane dla istotnych usług, oraz drugi o nazwie `a-noness`, w którym zbieramy dane o usługach nieistotnych. Następnie dodajemy reguły do naszego łańcucha przekazywania, który dopasowuje nasze istotne usługi i przechodzi do łańcucha `a-essent`, gdzie mamy tylko jedną regułę akceptującą wszystkie datagramy i je liczącą. Ostatnia reguła w naszym łańcuchu przekazywania to reguła, która przechodzi do łańcucha `a-noness`, w którym znów mamy jedną regułę przyjmującą i zliczającą wszystkie datagramy. Do reguły, która przechodzi do łańcucha `a-noness`, nie dotrze żadna z naszych istotnych usług, gdyż zostaną one zaakceptowane przez ich własny łańcuch. Rejestr istotnych i nieistotnych usług będzie dostępny w regułach tych łańcuchów. Opisaliśmy jeden ze sposobów, w jaki można liczyć ruch istotny i nieistotny – są oczywiście także inne. Nasza implementacja *iptables* wykorzystuje to samo podejście i przedstawia się tak:

```
# iptables -N a-essent
# iptables -N a-noness
# iptables -A a-essent -j ACCEPT
# iptables -A a-noness -j ACCEPT
# iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport ftp-data:ftp -j a-essent
# iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport smtp -j a-essent
# iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport www -j a-essent
# iptables -A FORWARD -j a-noness
```

Wygląda to dość prosto. Niestety występuje tu niewielki, ale nieunikniony problem przy próbie liczenia usług według typu. Pamiętasz, że w poprzednich rozdziałach mówiliśmy o roli, jaką w sieci TCP/IP odgrywa MTU. MTU definiuje największy datagram, jaki może zostać przesłany przez urządzenie sieciowe. Gdy datagram zosta-

nie odebrany przez ruter i jest większy niż MTU interfejsu, który musi go przetransmitować, ruter stosuje sztuczkę nazywaną *fragmentacją*. Ruter dzieli duży datagram na mniejsze fragmenty, nie większe jednak niż MTU interfejsu, a następnie je przesyła. Ruter tworzy nowe nagłówki, które umieszcza na początku każdego fragmentu. Zdalna maszyna używa ich do odtworzenia oryginalnych danych. Niestety w procesie fragmentacji port jest usuwany ze wszystkich fragmentów poza pierwszym. Oznacza to, że zliczanie IP nie jest w stanie poprawnie obsłużyć datagramów podzielonych na fragmenty. Może poprawnie policzyć tylko pierwszy fragment. Istnieje sztuczka wykonywana przez *ipfwadm*, która pozwala zliczać dalsze fragmenty, mimo że nie jesteśmy w stanie dowiedzieć się dokładnie, z jakiego portu pochodzą. Wcześniejsze wersje oprogramowania zliczającego dla Linuksa przypisywały fragmentom fałszywy numer portu 0xFFFF, który pozwalał liczyć. Aby mieć pewność, że uwzględniamy drugi i dalsze fragmenty, możemy użyć następującej reguły:

```
# ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 0xFFFF
```

Implementacja łańcuchów IP oferowała nieco bardziej wyrafinowane rozwiązanie, ale wynik był podobny. W przypadku polecenia *ipchains* mogliśmy użyć:

```
# ipchains -A forward -i ppp0 -p tcp -f
```

a w przypadku *iptables*:

```
# iptables -A FORWARD -i ppp0 -m tcp -p tcp -f
```

Nie pokaże nam to, jaki był oryginalny port danych, ale przynajmniej będziemy w stanie stwierdzić, ile danych zostało podzielonych na fragmenty i policzyć generowany przez nie ruch.

Podczas kompilacji jąder serii 2.2 możesz wybrać opcję, która usuwa cały ten problem, jeżeli twój komputer z Linuksem działa jako pojedynczy punkt dostępu do sieci. Jeżeli w czasie kompilacji jądra włączysz opcję *IP: always defragment*, wszystkie odebrane datagramy będą składane przez ruter linuksowy, zanim zostaną przerutowane i ponownie wysłane. Ta operacja jest realizowana przed filtrowaniem na firewallu i oprogramowanie zliczające widzi datagram tak, jakby nie był podzielony na fragmenty. W jądrach 2.4 musisz skompilować i załadować moduł *netfilter forward-fragment*.

Zliczanie datagramów ICMP

Protokół ICMP nie używa numerów portów usługi i dlatego nieco trudniej jest zbierać szczegółowe dane na jego temat. ICMP wykorzystuje różnego typu datagramy. Wiele z nich jest nieszkodliwych i normalnych, natomiast niektóre powinny pojawiać się tylko w pewnych okolicznościach. Czasami ludzie, którzy mają zbyt wiele czasu, próbują złośliwie zakłócić użytkownikom dostęp do sieci, generując dużą liczbę komunikatów ICMP. Powszechnie nazywa się to *zalaniem pingami* (ang. *ping flooding*). Choć za pomocą mechanizmu zliczania ruchu IP nie można zrobić nic, by zapobiec temu problemowi (choć może tu pomóc firewall IP!), możemy przynajmniej stworzyć reguły, które pokażą nam, czy ktoś próbuje takiego działania.

ICMP nie używa portów, tak jak TCP czy UDP. Natomiast ma różne typy komunikatów. Możesz stworzyć reguły liczące każdy typ komunikatu ICMP. W tym celu umieszczamy komunikat ICMP i numer typu w polu portu polecenia zliczające go *ipfwadm*. Typy komunikatów ICMP wymieniliśmy w podrozdziale *Typy datagramów ICMP*, a więc zajrzyj tam, jeżeli chcesz je sobie przypomnieć.

Reguła liczenia ruchu IP zbierająca informacje o liczbie pingów wysyłanych do naszej maszyny i z niej generowanych może wyglądać tak:

```
# ipfwadm -A both -a -P icmp -S 0/0 8
# ipfwadm -A both -a -P icmp -S 0/0 0
# ipfwadm -A both -a -P icmp -S 0/0 0xff
```

lub w przypadku *ipchains* tak:

```
# ipchains -A forward -p icmp -s 0/0 8
# ipchains -A forward -p icmp -s 0/0 0
# ipchains -A forward -p icmp -s 0/0 -f
```

lub w przypadku *iptables* tak:

```
# iptables -A FORWARD -m icmp -p icmp --sports echo-request
# iptables -A FORWARD -m icmp -p icmp --sports echo-reply
# iptables -A FORWARD -m icmp -p icmp -f
```

Pierwsza reguła zbiera informacje o datagramach „ICMP Echo Request" (żądania ping), a druga zbiera informacje o „ICMP Echo Reply" (odpowiedzi na ping). Trzecia reguła zbiera informacje o fragmentach datagramów ICMP. Jest to sztuczka podobna do opisanej w przypadku podzielonych na fragmenty datagramów TCP i UDP.

Jeżeli w swoich regułach podasz adres źródłowy i docelowy, możesz wyśledzić, skąd pochodzą pingi, z sieci wewnętrznej czy zewnętrznej. Gdy już ustalisz, skąd przychodzą szkodliwe datagramy, możesz rozważyć, czy chcesz umieścić regułę firewalla, która będzie zapobiegała ich przyjmowaniu, czy też podjąć jakieś inne działania, jak skontaktowanie się z właścicielem sieci, z której one pochodzą i powiadomienie go o problemie, a może nawet wytoczenie sprawy, jeżeli działanie było szczególnie złośliwe.

Zliczanie według protokołu

Wyobraźmy sobie teraz, że chcemy wiedzieć, jak duży ruch na naszym łączu generują protokoły TCP, UDP i ICMP. Użyjemy do tego celu następujących reguł:

```
# ipfwadm -A both -a -W ppp0 -P tcp -D 0/0
# ipfwadm -A both -a -W ppp0 -p udp -D 0/0
# ipfwadm -A both -a -W ppp0 -p icmp -D 0/0
```

lub:

```
# ipchains -A forward -i ppp0 -p tcp -d 0/0
# ipchains -A forward -i ppp0 -p udp -d 0/0
# ipchains -A forward -i ppp0 -p icmp -d 0/0
```

lub:

```
# iptables -A FORWARD -i ppp0 -m tcp -p tcp
# iptables -A FORWARD -o ppp0 -m tcp -p tcp
# iptables -A FORWARD -i ppp0 -m udp -p udp
# iptables -A FORWARD -o ppp0 -m udp -p udp
```

```
# iptables -A FORWARD -i ppp0 -m icmp -p icmp
# iptables -A FORWARD -o ppp0 -m icmp -p icmp
```

Zgodnie z regułami, cały ruch przechodzący przez interfejs `ppp0` będzie analizowany pod kątem ustalenia, czy jest to ruch TCP, UDP czy ICMP i będą uaktualniane odpowiednie liczniki dla każdego z protokołów. W przykładzie dla polecenia *iptables* ruch przychodzący jest oddzielany od wychodzącego, gdyż wymaga tego składnia.

Wykorzystywanie wyników zliczania ruchu IP

Bardzo dobrze, że zbieramy informacje, ale jak zobaczyć wynik? Aby obejrzeć zebrane dane o ruchu i skonfigurowane reguły zliczające, odwołujemy się do poleceń konfiguracyjnych firewalla, prosząc je o pokazanie listy reguł. W wyniku są pokazywane liczniki pakietów i bajtów dla każdej z naszych reguł.

Polecenia *ipfwadm*, *ipchains* i *iptables* różnie obsługują zebrane dane, a więc musimy je pokazać niezależnie.

Oglądanie danych za pomocą ipfwadm

Najprostszym sposobem na obejrzenie danych o ruchu za pomocą polecenia *ipfwadm* jest użycie go w następujący sposób:

```
# ipfwadm -A -l
IP accounting rules
 pkts bytes dir prot source              destination    ports
 9833 2345K i/o all  172.16.3.0/24       anywhere       n/a
56527   33M i/o all  172.16.4.0/24       anywhere       n/a
```

Widać tu liczbę pakietów wysłanych w obie strony. Gdybyśmy użyli opcji *–e* do pokazania wyniku w bogatszej postaci (nie pokazujemy tu taj, gdyż nie zmieściłby się na stronie), musielibyśmy podać również odpowiednie opcje i nazwy interfejsów. Większość pól tego wyniku jest oczywista, ale poniższe mogą wymagać wyjaśnienia:

dir
: Kierunek, którego dotyczy reguła. Możliwe wartości to `in`, `out` lub `i/o`, co oznacza oba kierunki.

prot
: Protokoły, których dotyczą reguły.

opt
: Zakodowana postać opcji, których używamy w wywołaniu *ipfwadm*.

ifname
: Nazwa interfejsu, którego dotyczy reguła.

ifaddress
: Adres interfejsu, którego dotyczy reguła.

Domyślnie *ipfwadm* wyświetla liczniki pakietów i bajtów w skróconej postaci, czyli zaokrąglone do najbliższego tysiąca (K) lub miliona (M). Możemy spowodować, by wyniki były wyświetlane bez zaokrąglania. Robi się to tak:

```
# ipfwadm -A -l -e -x
```

Oglądanie danych za pomocą ipchains

Polecenie *ipchains* nie wyświetli zebranych danych (liczników pakietów i bajtów), dopóki nie podamy argumentu *–v*. Najprostszy sposób na obejrzenie danych za pomocą *ipchains* jest następujący:

```
# ipchains -L -v
```

Znów, tak jak w *ipfwadm*, używając trybu rozszerzonego wyniku, możemy wyświetlić liczniki pakietów i bajtów w dokładnych jednostkach. *ipchains* do tego celu wykorzystuje argument `-x`:

```
# ipchains -L -v -x
```

Oglądanie danych za pomocą iptables

Polecenie *iptables* zachowuje się bardzo podobnie jak *ipchains*. Znów musimy użyć opcji `-v`, gdy chcemy obejrzeć nasze liczniki. Aby obejrzeć zebrane dane o ruchu, piszemy:

```
# iptables -L -v
```

Tak jak w poleceniu *ipchains*, możesz użyć argumentu `-x` do obejrzenia wyniku w rozszerzonej wersji, z liczbami w dokładnej postaci.

Zerowanie liczników

Liczniki zliczające ruch IP przepełnią się, jeżeli pozostawisz je włączone na dłuższy czas. Gdy się przepełnią, będziesz miał trudności w ustaleniu ich rzeczywistej wartości. Aby uniknąć tego problemu, powinieneś co jakiś czas odczytywać zebrane dane, zapisywać je, a następnie zerować liczniki, by ponownie rozpocząć zbieranie informacji o ruchu przez następny okres zliczeniowy.

Polecenia *ipfwadm* i *ipchains* udostępniają prosty sposób na zerowanie liczników:

```
# ipfwadm -A -z
```

lub:

```
# ipchains -Z
```

lub:

```
# iptables -Z
```

Możesz także połączyć wyświetlanie i zerowanie, by mieć pewność, że w międzyczasie dane się nie zagubią:

```
# ipfwadm -A -l -z
```

lub:

```
# ipchains -L -Z
```

lub:

```
# iptables -L -Z -v
```

Polecenia powyższe najpierw pokażą dane o ruchu, a następnie natychmiast wyzerują liczniki i rozpoczną zliczanie od nowa. Jeżeli chcesz zbierać i wykorzystywać informacje regularnie, prawdopodobnie umieścisz to polecenie w skrypcie uruchamianym co jakiś czas przez polecenie *cron*. Skrypt ten zapisuje wynik i gdzieś go zachowuje.

Usuwanie zestawów reguł

Ostatnie polecenie, które może być przydatne, pozwala ci usunąć wszystkie skonfigurowane wcześniej reguły zliczające IP. Jest najbardziej przydatne, gdy chcesz radykalnie zmienić zestaw reguł bez ponownego uruchamiania komputera.

Argument `-f` w połączeniu z poleceniem *ipfwadm* wyrzuci wszystkie reguły danego typu. *ipchains* obsługuje argument `-F`, który robi to samo:

```
# ipfwadm -A -f
```

lub:

```
# ipchains -F
```

lub:

```
# iptables -F
```

Powyższe polecenia powodują usunięcie wszystkich skonfigurowanych reguł zliczania ruchu IP, dzięki czemu nie tracisz czasu na usuwanie ich pojedynczo. Zauważ, że takie usuwanie reguł w przypadku *ipchains* nie powoduje usunięcia żadnego z łańcuchów zdefiniowanych przez użytkownika, a usuwa tylko zawarte w nich reguły.

Bierne zbieranie danych o ruchu

Ostatnia sztuczka, którą warto poznać: jeżeli twój Linux jest podłączony do Ethernetu, możesz zastosować reguły liczenia ruchu do wszystkich danych z twojego segmentu, a nie tylko do tych, które są przez niego przesyłane lub do niego adresowane. Twoja maszyna może biernie podsłuchiwać wszystkie dane przesyłane w segmencie sieci, do którego jest podłączona i je zliczać.

Powinieneś najpierw wyłączyć przekazywanie IP na twoim komputerze z Linuksem, tak by nie próbował on rutować odebranych datagramów*. W jądrach 2.0.36 i 2.2 robi się to za pomocą:

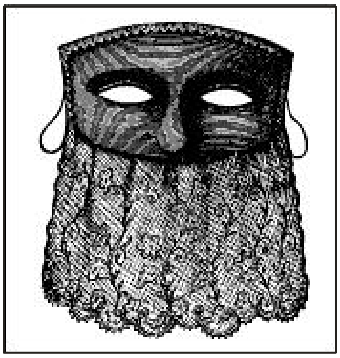
```
# echo 0>/proc/sys/net/ipv4/ip_forward
```

Następnie za pomocą polecenia *ifconfig*, powinieneś przejść na swojej karcie Ethernet do trybu przechwytywania (ang. *promiscuous*). Teraz możesz stworzyć reguły liczenia ruchu pozwalające ci zbierać informacje o datagramach przesyłanych w segmencie Ethernet bez włączania rutingu na swoim Linuksie.

* Nie jest to dobre, jeżeli twój Linux działa jako ruter. Jeżeli wyłączysz przekazywanie IP, maszyna przestanie rutować pakiety! Rób to tylko na komputerach z jednym fizycznym interfejsem sieciowym.

11

Maskowanie IP i translacja adresów sieciowych



Nie musisz mieć dobrej pamięci, by pamiętać czasy, gdy tylko duże instytucje mogły pozwolić sobie na połączenie wielu komputerów w sieć LAN. Obecnie technologia sieciowa jest na tyle tania, że możemy zaobserwować dwa zjawiska. Po pierwsze, sieci LAN są teraz powszechne, nawet dostępne w gospodarstwach domowych. Wielu użytkowników Linuksa ma po kilka komputerów połączonych siecią Ethernet. Po drugie, zasoby sieciowe, szczególnie adresy IP, kończą się i choć przyzwyczailiśmy się, że są za darmo, obecnie coraz częściej są kupowane i sprzedawane.

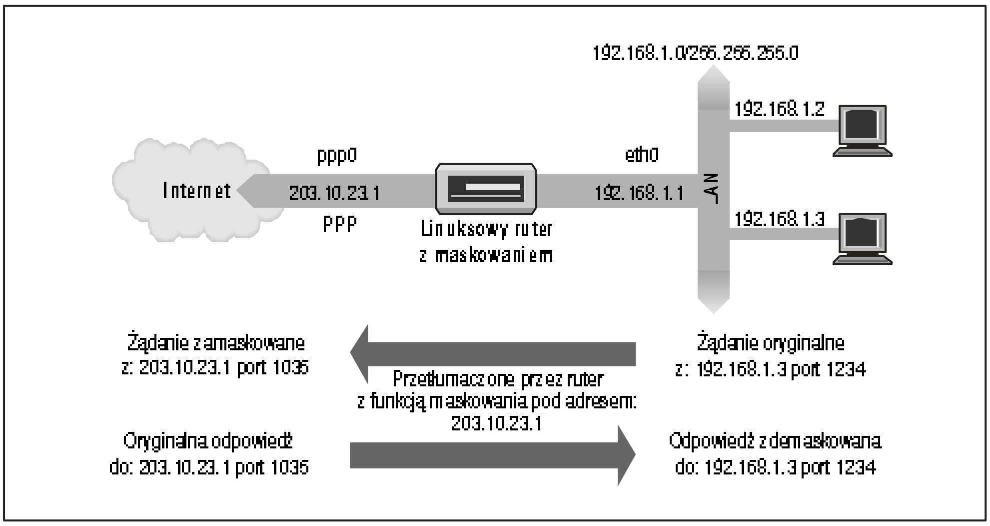
Większość posiadaczy sieci LAN zwykle chce także mieć połączenie z Internetem, wykorzystywane przez każdy komputer w sieci. Reguły rutingu IP są dosyć sztywne, jeśli chodzi o działanie w takiej sytuacji. Tradycyjne rozwiązania tego problemu zakładają uzyskanie adresu IP dla sieci, być może klasy C w przypadku mniejszych ośrodków, i przypisanie adresu każdemu hostowi sieci LAN, a następnie podłączenie sieci LAN do Internetu przez ruter.

W skomercjalizowanych środowiskach internetowych jest to dość droga propozycja. Po pierwsze, musiałbyś zapłacić za przypisanie twojej sieci adresów IP. Po drugie, musiałbyś zapłacić dostawcy usług internetowych za przywilej posiadania odpowiedniego rutera dla twojej sieci, dzięki któremu reszta Internetu wiedziałaby, jak się do niej dostać. Może być to wciąż praktyczne rozwiązanie dla firm, ale właściciele domowych instalacji zwykle nie są w stanie udźwignąć jego kosztów.

Na szczęście Linux oferuje inne rozwiązanie tego problemu. Rozwiązanie to wymaga elementu z grupy zaawansowanych funkcji sieciowych, tak zwanej *translacji adresów sieciowych* (*Network Address Translation* – NAT). Jest to proces modyfikacji adresów sieciowych zawartych w nagłówkach datagramu, który zachodzi w czasie ich przesyłania. Na początku może ci się to wydawać dość dziwne, ale zobaczysz wkrótce, że jest to idealne rozwiązanie opisywanego problemu i wiele osób z niego korzysta. Maskowanie IP (ang. *IP masquerading*) to nazwa nadana jednej z odmian translacji adresów sieciowych, która pozwala hostom z adresem sieci prywatnej przedstawić się w Internecie pod jednym publicznym adresem IP.

Maskowanie IP daje możliwość używania adresu IP z sieci prywatnej (zarezerwowanego) w twojej sieci LAN, a twój ruter oparty na Linuksie wykonuje w czasie rzeczywistym pewne inteligentne tłumaczenie adresów IP i portów. Gdy ruter odbierze datagram od komputera z sieci LAN, sprawdza, czy jest to datagram typu „TCP", „UDP", „ICMP" itp. i modyfikuje go tak, że wygląda on jakby był wygenerowany przez sam ruter (ruter pamięta, że to zrobił). Następnie ruter wysyła datagram do Internetu, nadając mu jeden adres IP. Gdy host docelowy odbierze datagram, uwierzy, że przyszedł on z rutera i wyśle w odpowiedzi datagramy na jego adres. Gdy ruter linuksowy z włączonym maskowaniem odbierze datagram przez swoje połączenie sieciowe, zajrzy do swojej tablicy aktywnych, maskowanych połączeń, by zobaczyć, czy datagram rzeczywiście należy do komputera w sieci LAN. Jeżeli tak, odwróci dokonaną przez siebie wcześniej modyfikację oraz wyśle datagram temu komputerowi.

Prosty przykład takiego działania pokazano na rysunku 11-1.



Rysunek 11-1. Typowa konfiguracja maskowania IP

Mamy małą sieć Ethernet wykorzystującą jeden z zarezerwowanych adresów sieci. W sieci jest linuksowy ruter z funkcją maskowania, dający dostęp do Internetu. Jedna ze stacji roboczych w sieci (192.168.1.3) chce połączyć się z hostem zdalnym o adresie 209.1.106.178 na porcie 8888. Stacja robocza kieruje swój datagram do rutera, który stwierdza, że żądanie połączenia wymaga maskowania. Przyjmuje datagram i alokuje port (1035), zastępuje adres i port hosta swoimi własnymi i przesyła datagram do hosta docelowego. Docelowy host myśli, że otrzymał żądanie połączenia od rutera linuksowego z włączonym maskowaniem i generuje datagram z odpowiedzią. Otrzymawszy datagram, ruter znajduje powiązanie w tablicy maskowania i odwraca operacje wykonane na wychodzącym datagramie. Następnie przesyła odpowiedź do hosta, od którego pierwotnie wyszedł datagram.

Lokalny host myśli, że komunikuje się bezpośrednio z hostem zdalnym. Zdalny host nic nie wie o hoście lokalnym, a myśli, że połączenie na wiązu je z ruterem. Ruter z maskowaniem wie, że te dwa hosty komunikują się ze sobą, jakich portów używają i dokonuje translacji adresów i portów wymaganej do uzyskania takiej komunikacji.

Może to wydawać się nieco zagmatwane, ale działa i łatwo się konfiguruje. Nie przejmuj się więc, jeżeli nie rozumiesz jeszcze szczegółów.

Skutki uboczne i dodatkowe korzyści

Funkcji maskowania IP towarzyszą pewne skutki uboczne, z których niektóre są użyteczne, a inne mogą przeszkadzać.

Po pierwsze, żaden z hostów sieci, która jest obsługiwana przez ruter maskujący, nie jest widziany bezpośrednio, dzięki czemu potrzebujesz tylko jednego poprawnego i rutowalnego adresu IP, aby wszystkie hosty mogły łączyć się z Internetem. Ma to taką wadę, że żaden z hostów nie jest widoczny z Internetu i nie możesz się do niego podłączyć bezpośrednio. Jedynym widocznym hostem w maskowanej sieci jest sama maszyna maskująca. Jest to istotne, gdy rozważasz usługi, takie jak poczta czy FTP. Pomaga ustalić, jakie usługi powinny być udostępniane przez ruter maskujący, a jakie powinny być udostępniane przez proxy lub traktowane w inny, szczególny sposób.

Po drugie, ponieważ żaden z maskowanych hostów nie jest widoczny, są one w pewnym sensie zabezpieczone przed atakami z zewnątrz. Za pewne upraszcza to konfigurowanie firewalla na hoście maskującym. Możesz się nawet zastanawiać, czy firewall jest w ogóle potrzebny. Nie powinieneś jednak zbytnio ufać maskowaniu. Cała twoja sieć jest tylko tak zabezpieczona jak host maskujący, a więc powinieneś użyć firewalla, jeżeli bezpieczeństwo jest dla ciebie istotne.

Po trzecie, maskowanie IP będzie miało pewien wpływ na wydajność sieci. W typowych konfiguracjach prawdopodobnie będzie to ledwo zauważalne. Gdybyś jednak miał wiele aktywnych, zamaskowanych sesji, mógłbyś zauważyć, że przetwarzanie realizowane na maszynie maskującej zaczyna wpływać na przepustowość sieci. Maskowanie IP wymaga wykonania sporej pracy dla każdego datagramu, porównywalnej z typowym rutingiem. Jeżeli planowałeś powierzyć komputerowi 386SX16 obsługę maskowania dla komutowanego łącza do Internetu, może to wystarczyć, ale nie spodziewaj się zbyt wiele, jeżeli zdecydujesz się go użyć jako rutera w sieci korporacyjnej działającej z prędkościami sieci Ethernet.

Pewne usługi sieciowe nie będą działały przy włączonej funkcji maskowania lub będą wymagały wsparcia. Zwykle są to usługi, które opierają się na przychodzących sesjach, takie jak bezpośrednie kanały komunikacyjne (*Direct Communications Channels* – DCC) w IRC-u czy pewne typy usług grupowych wideo i audio. Dla niektórych z nich stworzono specjalne moduły jądra po zwalające problem rozwiązać; za chwilę o tym powiemy. Może się jednak tak zdarzyć, że nie znajdziesz rozwiązania, a więc bądź ostrożny, gdyż nie zawsze da się zastosować maskowanie.

Konfigurowanie jądra do maskowania IP

Aby użyć funkcji maskowania IP, twoje jądro musi zostać skompilowane z jej obsługą. W czasie konfigurowania jądra serii 2.2 musisz wybrać następujące opcje:

```
Networking options --->
        [*] Network firewalls
        [*] TCP/IP networking
        [*] IP: firewalling
        [*] IP: masquerading
        --- Protocol-specific masquerading support will be built as modules.
        [*] IP: inautofw masq support
        [*] IP: ICMP masquerading
```

Zauważ, że obsługa maskowania jest dostępna tylko jako moduł jądra. Oznacza to, że w czasie kompilacji jądra musisz pamiętać o wykonaniu „`make modules`" poza zwykłym „`make zImage`".

Jądra serii 2.4 nie posiadają obsługi maskowania IP jako opcji wybieranej w czasie kompilacji jądra. Natomiast powinieneś wybrać opcję filtrowania pakietów sieciowych:

```
Networking options --->
    [M] Network packet filtering (replaces ipchains)
```

W czasie kompilacji jąder serii 2.2 powstaje szereg modułów pomocniczych, specyficznych dla protokołów. Niektóre protokoły rozpoczynają działanie od wysyłania żądań na jednym porcie, a potem oczekują na połączenia przychodzące na innym porcie. Protokoły takie zwykle nie mogą być maskowane, gdyż nie ma innego sposobu na powiązanie drugiego połączenia z pierwszym, jak powiązanie ich wewnątrz samych protokołów. Moduły pomocnicze to właśnie robią. W praktyce zaglądają do środka datagramów i umożliwiają działanie maskowania z niektórych protokołów, co w innej sytuacji byłoby niemożliwe. Obsługiwane protokoły to:

Moduł	*Protokół*
ip_masq_ftp	FTP
ip_masq_irc	IRC
ip_masq_raudio	RealAudio
ip_masq_cuseeme	CU-See-Me
ip_masq_vdolive	For VDO Live
ip_masq_quake	IdSoftwareQuake

Aby uruchomić te moduły, musisz je ręcznie załadować za pomocą polecenia *insmod*. Zauważ, że moduły te nie mogą być ładowane przez demona *kerneld*. Każdy z modułów przyjmuje argument, który określa, na jakich portach moduł będzie nasłuchiwał. Moduł RealAudio mógłbyś załadować w następująco**.

```
# insmod ip_masq_raudio.o ports=7070,7071,7072
```

Numery portów zależą od protokołu. Dokument mini-HOWTO o maskowaniu IP napisany przez Ambrose Au, podaje więcej szczegółów na temat modułów maskowania IP i omawia, jak je konfigurować**.

---

* RealAudio jest znakiem towarowym Progressive Networks Corporation.

** Z Ambrose możesz się skontaktować pod adresem *ambrose@writeme.com*.

Pakiet *netfilter* zawiera moduły działające podobnie. Na przykład, aby udostępnić połączenie śledzące sesje FTP, powinieneś załadować moduły *ip_conntrack_ftp* i *ip_nat_ftp.o* i użyć ich.

Konfigurowanie maskowania IP

Jeżeli już przeczytałeś rozdziały na temat firewalla i liczenia ruchu IP, nie będziesz zaskoczony, że do konfigurowania reguł maskowania IP są używane również polecenia *ipfwadm*, *ipchains* i *iptables*.

Reguły maskowania są szczególną klasą reguł filtrujących. Możliwe jest maskowanie jedynie tych datagramów, które są odbierane na jednym interfejsie i rutowane do innego interfejsu. W celu utworzenia reguły maskowania musisz skonstruować regułę bardzo podobną do reguły przekazywania dla firewalla, ale z wykorzystaniem specjalnych opcji, które mówią jądru, by maskowało datagram. Polecenie *ipfwadm* wykorzystuje opcję *–m*, *ipchains* opcję `-j MASQ`, a *iptables* opcję `-j MASQUERADE`, by pokazać, że datagram pasujący do reguły powinien być zamaskowany.

Spójrzmy na przykład. Student informatyki z uniwersytetu Groucho Marx ma w domu kilka komputerów połączonych w sieć Ethernet. Zdecydował się na użycie jednego z prywatnych, zarezerwowanych adresów sieci. Mieszka razem z innymi studentami, którzy też chcą mieć dostęp do Internetu. Ponieważ ich warunki życiowe są dość skromne, nie mogą pozwolić sobie na stałe połączenie z Internetem, a więc używają zwykłego, komutowanego połączenia PPP. Wszyscy chcieliby współdzielić łącze, by pogadać na IRC-u, pooglądać strony WWW czy pobrać pliki przez FTP bezpośrednio na swoje komputery – maskowanie IP jest tu dobrym rozwiązaniem.

Student konfiguruje najpierw komputer z Linuksem obsługujący łącze komutowane i działający jako ruter dla sieci LAN. Adres IP, jakiego używa przy dzwonieniu, nie jest istotny. Konfiguruje ruter z maskowaniem IP i używa jednego z adresów sieci prywatnej dla swojej sieci LAN: `192.168.1.0`. Zapewnia każdemu hostowi w sieci LAN ustawienie domyślnego rutingu tak, by wskazywał na ruter linuksowy.

Poniższe polecenia *ipfwadm* wystarczą, by maskowanie zadziałało w takiej konfiguracji:

```
# ipfwadm -F -p deny
# ipfwadm -F -a accept -m -S 192.168.1.0/24 -D 0/0
```

lub w przypadku *ipchains*:

```
# ipchains -P forward -j deny
# ipchains -A forward -s 192.168.1.0/24 -d 0/0 -j MASQ
```

lub w przypadku *iptables*:

```
# iptables -t nat -P POSTROUTING DROP
# iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

Teraz, gdy tylko jakiś z hostów z sieci LAN spróbuje połączyć się z usługą na hoście zdalnym, jego datagramy zostaną automatycznie zamaskowane przez ruter z Linuksem. Pierwsza reguła w każdym z przykładów zapobiega przed rutowaniem wszelkich innych datagramów i zapewnia bezpieczeństwo.

Aby zobaczyć właśnie utworzoną listę reguł maskowania, użyj argumentu `-l` w poleceniu *ipfwadm* zgodnie z tym, co opisaliśmy, omawiając firewalle.

Oto, jak należy zapisać to polecenie:

```
# ipfwadm -F -l -e
```

W wyniku otrzymujemy:

```
# ipfwadm -F -l -e
IP firewall forward rules, default policy: accept
 pkts bytes type  prot opt tosa tosx ifname ifaddress ...
    0     0 acc/m all ---- 0xFF 0x00 any    any       ...
```

Symbol „`/m`" w wyniku pokazuje właśnie regułę maskowania.

Aby zobaczyć listę reguł maskowania w przypadku polecenia *ipchains*, użyj argumentu *–L*. Wynik będzie następujący:

```
# ipchains -L
Chain input (policy ACCEPT):
Chain forward (policy ACCEPT):
target     prot   opt    source            destination      ports
MASQ       all  ------  192.168.1.0/24    anywhere         n/a
Chain output (policy ACCEPT):
```

Wszelkie reguły, w których pole `target` ma wartość `MASQ`, to reguły maskowania.

No i aby zobaczyć reguły za pomocą *iptables*, musisz użyć czegoś takiego:

```
# iptables -t net -L
Chain PREROUTING (policy ACCEPT)
target     prot   opt    source      destination

Chain POSTROUTING (policy DROP)
target     prot   opt    source      destination
MASQUERADE  all   --   anywhere      anywhere              MASQUERADE

Chain OUTPUT (policy ACCEPT)
target     prot   opt    source      destination
```

Znów, reguły maskowania są oznaczone słowem `MASQUERADE` w polu `target`.

Ustawianie parametrów czasowych dla maskowania IP

Gdy jest nawiązywane każde nowe połączenie, oprogramowanie maskowania IP tworzy w pamięci powiązanie pomiędzy każdym z hostów w nim uczestniczących. Możesz obejrzeć te powiązania w dowolnej chwili, zaglądając do pliku */proc/net/ip_masquerade*. Powiązania wygasną po pewnym czasie nieaktywności.

Za pomocą polecenia *ipfwadm* możesz ustawić wartości czasu ich wygaśnięcia. Ogólna składnia jest następująca:

```
ipfwadm -M -s <tcp> <tcpfin> <udp>
```

i dla polecenia *ipchains* jest tak:

```
ipchains -M -S <tcp> <tcpfin> <udp>
```

Implementacja *iptables* wykorzystuje liczniki czasowe o dużo większej wartości i nie pozwala na ich ustawianie.

Każda z tych wartości określa licznik czasu używany przez oprogramowanie maskowania IP i jest wyrażona w sekundach. Poniższa tabela podsumowuje liczniki i ich znaczenie:

Nazwa	*Opis*
tcp	Czas przerwy w sesji TCP. Mówi, jak długo połączenie TCP może pozostawać jałowe, za nim powiązanie zostanie usunięte.
tcpfin	Czas oczekiwania TCP po napotkaniu FIN. Mówi, jak długo powiązanie pozostanie w tablicy po rozłączeniu połączenia TCP.
udp	Czas przerwy w sesji UDP. Mówi, jak długo połączenie UDP może pozostawać jałowe, za nim powiązanie zostanie usunięte.

Obsługiwanie przeszukiwania serwerów nazw

Obsługiwanie przeszukiwania serwerów nazw z hostów w sieci z maskowaniem IP zawsze stanowiło problem. Istnieją dwa sposoby na dostosowanie DNS-u do środowiska z maskowaniem. Możesz powiedzieć każdemu z hostów, żeby używał tego samego DNS-u co ruter i niech maskowanie IP obsługuje ich zapytania DNS. Albo możesz uruchomić linuksowy serwer pamięci podręcznej i ustawić go tak, by każdy z hostów używał go jako swojego DNS-u. Jest to bardziej agresywne działanie, ale o tyle korzystniejsze, że redukuje rozmiar ruchu DNS przesyłanego do łącza internetowego i będzie nieco szybsze, ponieważ większość zapytań będzie obsługiwana z pamięci podręcznej. Wadą tej konfiguracji jest to, że jest bardziej skomplikowana. Podrozdział *Konfiguracja na med jak serwera pamięci podręcznej* w rozdziale 6 opisuje przebieg konfiguracji takiego serwera.

Więcej na temat translacjiadresów sieciowych

Oprogramowanie *netfilter* jest w stanie obsłużyć wiele różnych typów translacji adresów sieciowych. Maskowanie IP jest jednym z prostszych jego zastosowań.

Możliwe jest na przykład stworzenie takich reguł translacji, które będą tłumaczyć tylko pewne adresy lub zakresy adresów, a reszta pozostanie nietknięta lub będą tłumaczyć adresy na pule adresów, a nie na pojedyncze adresy, tak jak w maskowaniu. W praktyce możesz użyć polecenia *iptables* do stworzenia reguł translacji, które odwzorowują wszystko, wykorzystując połączenie różnych atrybutów, takich jak adres źródłowy, adres docelowy, typ protokołu, numer portu itp.

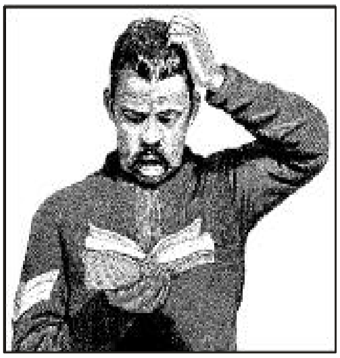
W dokumentacji *netfilter* tłumaczenie adresu źródłowego datagramu nazywa się `SNAT` (Source NAT). Tłumaczenie adresu docelowego datagramu jest nazywane `DNAT` (Destination NAT). Tłumaczenie portu TCP lub UDP jest znane pod pojęciem `REDIRECT`. `SNAT`, `DNAT` i `REDIRECT` to cele, których możesz używać w poleceniu *iptables* do tworzenia bardziej wyrafinowanych i skomplikowanych reguł.

Opisem odmian translacji adresów sieciowych można by wypełnić cały oddzielny rozdział*. Niestety my nie mamy tu dość miejsca. Jeżeli chcesz się dowiedzieć więcej o tym, jak używać translacji adresów sieciowych, powinieneś przeczytać *IPTABLES-HOWTO*.

* ... lub nawet całą książkę!

12

Ważne funkcje sieciowe



Po pomyślnym skonfigurowaniu IP i resolvera, warto przyjrzeć się usługom, które możesz udostępniać w sieci. Ten rozdział omawia konfigurację kilku prostych zastosowań sieci, między innymi serwera *inetd* i programów z rodziny *rlogin*. Krótko omówimy także interfejs zdalnego wywołania procedur (RPC – *Remote Procedure Call*), na którym oparte są usługi, takie jak sieciowy system plików (NFS – *Network File System*) i system informacji sieciowej (NIS – *Network Information System*). Jednak konfiguracja NFS-a i NIS-a są bardziej złożone i omówimy je w oddzielnych rozdziałach, podobnie jak pocztę elektroniczną oraz grupy dyskusyjne.

Oczywiście nie możemy w tej książce opisać wszystkich zastosowań sieci. Gdybyś chciał zainstalować coś, czego nie omawiamy, na przykład *talk*, *gopher* czy *http*, szukaj objaśnień na stronach podręcznika elektronicznego danego serwera.

Superserwer inetd

Programy udostępniające usługi przez sieć są nazywane *demonami* sieciowymi. Demon to program, który otwiera port (przeważnie jest to dobrze znany port usługi) i oczekuje na przychodzące na niego połączenia. Jeżeli takie nadejdzie, demon tworzy proces potomny przyjmujący połączenie, natomiast proces macierzysty dalej oczekuje na kolejne żądania. Mechanizm ten się sprawdza, ale ma kilka wad. Przynajmniej jedna instancja każdej możliwej usługi, którą chcesz udostępniać, musi być aktywna w pamięci przez cały czas. Ponadto procedury oprogramowania nasłuchującego go i obsługującego go port muszą być replikowane w każdym demonie sieciowym.

Aby działanie demona stało się bardziej efektywne, większość instalacji Uniksa uruchamia specjalnego demona sieciowego, którego możesz uznać za „superserwer". Demon ten tworzy gniazda w imieniu szeregu usług i nasłuchuje na nich wszystkich równocześnie. Gdy na którymś z tych gniazd zostanie odebrane przychodzące połączenie, superserwer przyjmuje je i uruchamia serwer dla określonego portu, przekazując gniazdo do obsłużenia przez proces potomny. Następnie powraca do nasłuchiwania.

Najpopularniejszy superserwer nosi nazwę *inetd* (z ang. *Internet Daemon*). Jest on uruchamiany w czasie startu systemu i przyjmuje listę usług do obsłużenia na podstawie pliku o nazwie */etc/inetd.conf*. Poza tymi usługami, istnieje szereg prostych usług realizowanych przez sam *inetd*, nazywanych *usługami wewnętrznymi*. Zaliczają się do nich *chargen*, generujący po prostu ciąg znaków, i *daytime*, który zwraca czas systemowy.

Wpis w tym pliku składa się z pojedynczego wiersza, w którym znajdują się z kolei następujące pola:

`usługa typ protokół wait użytkownik serwer wiersz-poleceń`

Każde z pól opisujemy poniżej:

`usługa`
: Zawiera nazwę usługi. Nazwa usługi musi być przetłumaczona na numer portu na podstawie pliku */etc/services*. Plik ten zostanie opisany dalej, w podrozdziale *Pliki services i protocols*.

`typ`
: Określa typ gniazda, czyli `stream` (dla protokołów zorientowanych połączeniowo) lub `dgram` (dla protokołów datagramowych). Usługi oparte na TCP powinny zawsze używać słowa `stream`, a usługi oparte na UDP, słowa `dgram`.

`protokół`
: Nazwa protokołu transportowego używanego przez usługę. Musi być to dopuszczalna nazwa protokołu znajdująca się w pliku *protocols*, opisanym dalej.

`wait`
: Ta opcja dotyczy tylko gniazd `dgram`. Może przyjmować wartość `wait` lub `nowait`. Jeżeli poda `nowait`, *inetd* uruchamia tylko pojedynczy serwer dla zadanego portu. W przeciwnym razie natychmiast po uruchomieniu serwera zaczyna nasłuchiwać na porcie.

: Jest to przydatna opcja dla serwerów „jednowątkowych”, które czytają wszystkie datagramy, dopóki przychodzą, a następnie kończą pracę. Większość serwerów RPC jest tego typu i powinny w tym polu zawierać słowo `wait`. Działające odwrotnie serwery – „wielowątkowe” – powodują nawet uruchamianie nieograniczonej liczby instancji. Takie serwery powinny mieć tu wpisane `nowait`.

: Gniazda `stream` powinny zawsze używać słowa `nowait`.

`użytkownik`
: Jest to ID użytkownika, który jest właścicielem procesu w czasie jego uruchamiania. Często będzie to użytkownik **root**, ale niektóre usługi mogą wykorzystywać inne konta. Dobrze jest stosować tu zasadę ograniczonego przywileju, która mówi, że nie powinieneś uruchamiać polecenia z prawami konta uprzywilejowanego, jeżeli program nie wymaga tego do poprawnego działania. Na przykład serwer grup dyskusyjnych NNTP działa jako **news**, natomiast usługi, które mogą potencjalnie zagrażać bezpieczeństwu (takie jak *tftp* czy *finger*) często są uruchamiane jako **nobody**.

`serwer`

Zawiera pełną ścieżkę do uruchamianego programu serwera. Wewnętrzne usługi są tu tajoznaczane słowem kluczowym `internal`.

`wiersz-polece□`

Jest to wiersz poleceń przekazywany do serwera. Rozpoczyna się od nazwy uruchamianego serwera i może zawierać wszelkie argumenty, które powinny być przekazane serwerowi. Jeżeli używasz wrappera TCP, podajesz tu tajpełną ścieżkę do serwera. W przeciwnym razie podajesz jedynie nazwę serwera, taką jaka pojawia się na liście procesów. Wkrótce powiemy o wrapperach TCP.

W przypadku usług wewnętrznych pole to jest puste.

Przykładowy plik *inetd.conf* został pokazany w przykładzie 12-1. Usługa *finger* jest zakomentowana, a więc nie jest dostępna. Często tak się robi ze względów bezpieczeństwa, ponieważ usługa ta może być używana przez osoby niepowołane do uzyskania nazw i innych szczegółów na temat użytkowników twojego systemu.

Przykład 12-1. Przykładowy plik /etc/inetd.conf

```
#
# us□ugi inetd
ftp      stream tcp nowait root /usr/sbin/ftpd    in.ftpd -l
telnet   stream tcp nowait root /usr/sbin/telnetd in.telnetd -b /etc/issue
#finger  stream tcp nowait bin  /usr/sbin/fingerd in.fingerd
#tftp    dgram  udp wait nobody /usr/sbin/tftpd   in.tftpd
#tftp    dgram  udp wait nobody /usr/sbin/tftpd   in.tftpd /boot/diskless
#login   stream tcp nowait root /usr/sbin/rlogind in.rlogind
#shell   stream tcp nowait root /usr/sbin/rshd    in.rshd
#exec    stream tcp nowait root /usr/sbin/rexecd  in.rexecd
#
#      us□ugi wewn□trzne inetd
#
daytime  stream tcp nowait root internal
daytime  dgram  udp nowait root internal
time     stream tcp nowait root internal
time     dgram  udp nowait root internal
echo     stream tcp nowait root internal
echo     dgram  udp nowait root internal
discard  stream tcp nowait root internal
discard  dgram  udp nowait root internal
chargen  stream tcp nowait root internal
chargen  dgram  udp nowait root internal
```

Demon *tftp* jest również zakomentowany. *tftp* implementuje *uproszczony protokół przesyłania plików* (*Trivial File Transfer Protocol* – TFTP), który pozwala każdemu, bez sprawdzania hasła, pobrać z systemu dowolny plik, który ma prawo do odczytu ustawione dla wszystkich. Jest to szczególnie niebezpieczne w przypadku pliku */etc/passwd*, a jeszcze bardziej, jeżeli nie używamy haseł typu shadow.

TFTP jest powszechnie stosowany przez klienty bezdyskowe i Xterminale do ładowania z serwera swojego kodu. Gdybyś z tego powodu musiał uruchomić *tftpd*, ogranicz dostęp tylko do tych katalogów, z których klienty będą odczytywać pliki. Będziesz musiał podać nazwy tych katalogów w wierszu poleceń *tftpd*. Pokazano to w drugim wierszu *tftp* w powyższym przykładzie.

Funkcja kontroli dostępu tcpd

Ponieważ udostępnianie komputera w sieci stwarza wiele zagrożeń dla bezpieczeństwa, aplikacje są tak projektowane, by bronić się przed pewnymi typami ataków. Niektóre funkcje bezpieczeństwa bywają jednak słabe (co pokazał internetowy robak RTM, który wykorzystał dziury w kilku programach, także w starych wersjach demona pocztowego *sendmail*) lub nie rozróżniają bezpiecznych hostów, od których żądanie jakiejś usługi można przyjąć, od niebezpiecznych hostów, których żądania powinny być odrzucone. Opisaliśmy już krótko usługi *finger* i *tftp*. Administrator sieci będzie chciał, aby jedynie zaufane hosty mogły z nich korzystać, co jest niemożliwe przy typowej konfiguracji, w której *inetd* udostępnia usługę wszystkim klientom lub nie udostępnia jej nikomu.

Przydatnym narzędziem do zarządzania dostępem ograniczonym do zadanego hosta jest *tcpd*, często nazywane demonem „wrappera"*. W przypadku usług TCP, które chcesz monitorować lub zabezpieczać, jest on wywoływany zamiast programu serwera. *tcpd* sprawdza, czy zdalny host ma prawo używać usługi i jeżeli test zakończy się poprawnie, uruchamia prawdziwy serwer. *tcpd* zapisuje również żądania do demona *syslog*. Zauważ, że demon ten nie obsługuje usług opartych na UDP.

Na przykład, aby „opakować" demona *finger*, musisz zmienić odpowiednią linię w pliku *inetd.conf* z takiej:

```
# demon finger wywo□ywany bezpo□rednio
finger    stream tcp nowait bin    /usr/sbin/fingerd in.fingerd
```

na taką:

```
# "opakowany" demon finger
finger    stream tcp nowait root   /usr/sbin/tcpd in.fingerd
```

Przy zupełnym braku kontroli dostępu będzie to wyglądało z punktu widzenia klienta tak jak zwykła konfiguracja *finger*, z tym wyjątkiem, że wszelkie żądania będą zapisywane przez funkcję o nazwie *sysloga auth*.

Kontrola dostępu jest realizowana przez dwa pliki: */etc/hosts.allow* i */etc/hosts.deny*. Zawierają one wpisy, które pozwalają na dostęp do pewnych usług i hostów lub go zabraniają. Gdy *tcpd* obsługuje żądanie usługi, na przykład *finger* od klienta z hosta o nazwie **biff.foobar.com**, przegląda pliki *hosts.allow* i *hosts.deny* (w tej kolejności) w poszukiwaniu wpisu pasującego zarówno do usługi, jak i do hosta. Jeżeli odpowiedni wpis zostanie znaleziony w pliku *hosts.allow*, dostęp jest udzielany, a *tcpd* nie sprawdza pliku *hosts.deny*. Jeżeli w pliku *hosts.allow* nie zostanie znaleziony wpis, ale zostanie on znaleziony w *hosts.deny*, żądanie jest odrzucane przez zamknięcie połączenia. Żądanie jest jednak akceptowane, jeżeli odpowiedni wpis nie zostanie znaleziony w żadnym z plików.

Wpisy w plikach dostępu wyglądają tak:

```
listaus□ug: listahostów [:polecenie]
```

* Autorem tego demona jest Wietse Venema, *wietse@wzv.win.tue.nl*.

`listausług` to lista nazw usług na podstawie pliku */etc/services* lub słowo kluczowe `ALL`. Aby pasowały wszystkie usługi poza *finger* i *tftp*, użyj `ALL EXCEPT finger, tftp`.

`listahostów` to lista nazw hostów, adresów IP lub słów kluczowych `ALL`, `LOCAL`, `UNKNOWN` lub `PARANOID`. `ALL` pasuje do dowolnego hosta, natomiast `LOCAL` tylko do hostów, które nie zawierają kropki*. Do `UNKNOWN` pasuje dowolny host, którego nazwy lub adresu nie znaleziono. Do `PARANOID` pasuje dowolny host, którego nazwa nie zamienia się na jego poprawny adres IP**. Do nazwy rozpoczynającej się od kropki pasują wszystkie hosty z domeny o tej nazwie. Na przykład do **.foobar.com** pasuje **biff.foobar.com**, ale nie pasuje **nurks.fredsville.com**. Do wzorca kończącego się kropką pasuje dowolny host, którego adres IP rozpoczyna się od podanego wzorca, a więc do **172.16** pasuje **172.16.32.0**, ale nie pasuje **172.15.9.1**. Wzorzec postaci `n.n.n.n/m.m.m.m` jest traktowany jak adres IP i maska sieci, a więc możemy zapisać poprzedni przykład jako **172.16.0.0/255.255.0.0**. Wreszcie dowolny wzorzec rozpoczynający się od znaku „/" pozwala na zadanie pliku, w którym zawarta będzie lista dopasowywanych wzorców nazw hostów lub adresów IP. Tak więc wzorzec wyglądający jak */var/access/trustedhosts* spowodowałby, że demon *tcpd* odczytałby plik, sprawdzając, czy jakieś zawarte w nim wiersze odpowiadają podłączającemu się hostowi.

Aby zakazać dostępu do usług *finger* i *tcpd* wszystkim oprócz hostów lokalnych, umieść następujący wpis w pliku */etc/hosts.deny* i pozostaw plik */etc/hosts.allow* pusty:

```
in.tftpd, in.fingerd: ALL EXCEPT LOCAL, .twoja.domena
```

Opcjonalne pole `polecenie` może zawierać polecenie wywoływane po dopasowaniu wpisu. Jest to przydatne do konfigurowania pułapek, które mogą ujawniać potencjalnych atakujących. Poniższy przykład tworzy plik logu, w którym są wpisywani użytkownicy i podłączające się hosty. Jeżeli host nie nazywa się **vlager.vbrew.com**, do nazwy zostanie dodany wynik polecenia *finger*:

```
in.ftpd: ALL EXCEPT LOCAL, .vbrew.com : \
      echo "request from %d@%h: >> /var/log/finger.log; \
      if [ %h != "vlager.vbrew.com:" ]; then \
          finger -l @%h >> /var/log/finger.log \
      fi
```

Argumenty `%h` i `%d` są rozwijane przez *tcpd* odpowiednio do nazwy hosta klienta i nazwy usługi. Szczegóły znajdziesz na stronie podręcznika elektronicznego *hosts_access(5)*.

* Zwykle tylko nazwy lokalne uzyskane z */etc/hosts* nie zawierają kropek.

** Chociaż nazwa sugeruje, że jest to warunek ekstremalny, słowo kluczowe `PARANOID` jest dobrą wartością domyślną, gdyż zabezpiecza przed złośliwymi hostami, które udają, że są kimś, kim nie są. Nie wszystkie wersje *tcpd* mają wkompilowaną obsługę `PARANOID`. Jeżeli twoja wersja nie ma, musisz przekompilować *tcpd*.

Pliki services i protocols

Numery portów, na których są udostępniane pewne „standardowe" usługi zostały zdefiniowane w dokumencie RFC *Assigned Numbers*. Aby programy serwera i klienta mogły konwertować nazwy usług na te numery, każdy host posiada przynajmniej część tej listy. Znajduje się ona w pliku o nazwie */etc/services*. Wpis wygląda tak:

```
us□uga   port/protokó□   [aliasy]
```

`Us□uga` oznacza tu tajna zwę usługi, a `port` definiuje port, na którym jest ona oferowana, natomiast `protokó□` określa używany protokół transportowy. To ostatnie pole przeważnie ma wartość `udp` lub `tcp`. Zdarza się, że usługa jest udostępniana nie tylko poprzez jeden protokół, oraz że na tym samym porcie udostępniane są różne usługi, jeżeli protokoły są różne. Pole `aliasy` pozwala określić alternatywne nazwy tej samej usługi.

Zwykle nie musisz zmieniać pliku *services*, który jest dostarczany wraz z oprogramowaniem sieciowym twojego Linuksa. Niemniej jednak w przykładzie 12-2 pokazujemy urywek tego pliku.

Przykład 12-2. Przykładowy plik /etc/services

```
# Plik services:
#
# dobrze znane us□ugi
echo         7/tcp                # Echo
echo         7/udp                #
discard      9/tcp sink null      # Discard
discard      9/udp sink null      #
daytime     13/tcp                # Daytime
daytime     13/udp                #
chargen     19/tcp ttytst source  # Generator znaków
chargen     19/udp ttytst source  #
ftp-data    20/tcp                # Protokó□ transmisji plików (dane)
ftp         21/tcp                # Protokó□ transmisji plików (sterowanie)
telnet      23/tcp                # Protokó□ wirtualnego terminala
smtp        25/tcp                # Prosty protokó□ przesy□ania poczty elektronicznej
nntp       119/tcp readnews       # Protokó□ przesy□ania wiadomo□ci w sieci USENET
#
# us□ugi UNIX
exec       512/tcp                # zdalne wykonywanie BSD
biff       512/udp comsat         # powiadomienie o poczcie
login      513/tcp                # zdalne logowanie
who        513/udp whod           # zdalne who i uptime
shell      514/tcp cmd            # zdalne polecenie, has□o nie jest u□ywane
syslog     514/udp                # zdalny syslog
printer    515/tcp spooler        # zdalne buforowanie drukowania
route      520/udp router routed  # protokó□ informacyjny rutowania
```

Zauważ, że usługa *echo* jest udostępniana na porcie 7 zarówno protokołowi TCP, jak i UDP, i że port 512 jest używany przez dwie różne usługi: zdalne wykonywanie (*rexec*) przez TCP i demona *COMSAT* na UDP, powiadamiającego użytkowników o nowej poczcie (zobacz *xbiff(1x)*).

Podobnie jak plik *services*, biblioteka sieciowa potrzebuje sposobu na przetłumaczenie nazw protokołów – na przykład używanych w pliku *services* – na numery proto-

kołów rozumiane przez warstwę IP na innych hostach. Dlatego poszukuje nazwy w pliku */etc/protocols*. Plik ten zawiera po jednym wpisie w wierszu, a każdy wpis podaje nazwę protokołu i odpowiadający jej numer. Prawdopodobieństwo robienia czegokolwiek z tym plikiem jest jeszcze mniejsze niż w przypadku */etc/services*. Przykładowy plik pokazujemy poniżej.

Przykład 12-3. Przykładowy plik /etc/protocols

```
#
# Protoko□y internetowe (IP)
#
ip      0      IP       # protokó□ internetowy, pseudonumer protoko□u
icmp    1      ICMP     # internetowy protokó□ komunikatów kontrolnych
igmp    2      IGMP     # protokó□ grupowy Internet
tcp     6      TCP      # protokó□ steruj□cy transmisj□
udp     17     UDP      # protokó□ datagramów u□ytkownika
raw     255    RAW      # interfejs RAW IP
```

Zdalne wywołanie procedur

Ogólny mechanizm aplikacji klient-serwer jest udostępniany przez RPC – pakiet *zdalnego wywołania procedur*. RPC powstał w firmie Sun Microsystems. Jest to zbiór narzędzi i funkcji bibliotecznych. Ważne aplikacje zbudowane w oparciu o RPC to NIS – system informacji sieciowej i NFS – sieciowy system plików. Oba zostaną opisane w tej książce, odpowiednio w rozdziałach 13, *System informacji sieciowej* i 14, *Sieciowy system plików*.

Serwer RPC zawiera zbiór procedur, które klient może wywoływać, wysyłając żądania RPC do serwera wraz z parametrami procedury. Serwer wywoła wskazaną procedurę w imieniu klienta i przekaże mu zwróconą wartość, jeżeli taka będzie. Aby uniezależnić się od maszyny, wszystkie dane wymieniane pomiędzy klientem i serwerem są konwertowane przez wysyłające go do formatu *zewnętrznej reprezentacji danych* (*External Data Representation* – XDR) i konwertowane z powrotem do lokalnej reprezentacji przez odbiorcę. RPC opiera się na standardowych gniazdach UDP i TCP przy przenoszeniu danych w formacie XDR do zdalnego hosta. Firma Sun udostępniła RPC na zasadach oprogramowania do publicznego rozpowszechniania. Opisano je w wielu RFC.

Czasem poprawki w aplikacji RPC są niekompatybilne z interfejsem wywołania procedur. Oczywiście zwykła zmiana serwera spowoduje awarię aplikacji, które wciąż oczekują pierwotnego działania. Dlatego programy RPC mają przypisane numery wersji, zwykle rozpoczynające się od 1, a potem, wraz z każdą nową wersją interfejsu RPC, licznik ten jest zwiększany. Często serwer może obsługiwać kilka wersji jednocześnie, a klienty w żądaniu wskazują numer wersji implementacji, której chcą używać.

Komunikacja pomiędzy serwerami i klientami RPC jest w pewnym sensie szczególna. Serwer RPC udostępnia jeden lub kilka zbiorów procedur. Każdy zestaw nazywa się *programem* i jest unikalnie identyfikowany przez *numer programu*. Lista odwzorowująca nazwy usług na numery programów zwykle znajduje się w pliku */etc/rpc*, którego fragment pokazano w przykładzie 12-4.

Przykład 12-4. Przykładowy plik /etc/rpc

```
#
# /etc/rpc - ró□ne us□ugi oparte na RPC
#
portmapper     100000  portmap sunrpc
rstatd         100001  rstat rstat_svc rup perfmeter
rusersd        100002  rusers
nfs            100003  nfsprog
ypserv         100004  ypprog
mountd         100005  mount showmount
ypbind         100007
walld          100008  rwall shutdown
yppasswdd      100009  yppasswd
bootparam      100026
ypupdated      100028  ypupdate
```

W sieciach TCP/IP autorzy RPC stanęli wobec problemu odwzorowania numerów programów na typowe usługi sieciowe. Zaprojektowali każdy serwer tak, by udostępniał porty TCP i UDP dla każdego programu i każdej wersji. Generalnie do wysyłania danych aplikacje RPC używają UDP, a TCP jedynie wtedy, gdy przesyłane dane nie mieszczą się w jednym datagramie UDP.

Oczywiście programy klientów muszą dopasować numer portu do numeru programu. Wykorzystanie w tym celu pliku konfiguracyjnego byłoby zbyt mało elastyczne. Ponieważ aplikacje RPC nie używają zarezerwowanych portów, nie ma gwarancji, że port, który pierwotnie był przeznaczony dla naszej aplikacji bazy danych, nie został zabrany przez inny proces. Dlatego aplikacje RPC biorą dostępny port i rejestrują go w specjalnym programie, nazywanym *demonem portmapper*. Portmapper działa jako usługa pośrednia dla wszystkich serwerów RPC funkcjonujących na danym komputerze. Klient, który chce skontaktować się z usługą o danym numerze programu, najpierw wysyła zapytanie do portmappera na danym hoście, a ten zwraca mu numery portów TCP i UDP, pod którymi usługa jest dostępna.

Niestety portmapper może odmówić działania już wtedy, gdy zostanie uszkodzony tylko jeden punkt, podobnie jak demon *inetd* dla standardowych usług Berkeley. Jednak ten przypadek jest jeszcze gorszy, gdyż jeżeli portmapper nie zadziała, wszystkie informacje o portach RPC zostaną stracone. Zwykle oznacza to, że będziesz musiał ponownie ręcznie uruchomić wszystkie serwery RPC lub całą maszynę.

W Linuksie portmapper nosi nazwę */sbin/portmap* lub czasem */usr/sbin/rpc.portmap*. Poza sprawdzeniem, czy jest on uruchamiany przez sieciowe skrypty startowe, nie wymaga żadnej konfiguracji.

Konfigurowanie zdalnego logowania i uruchamiania

Często bardzo przydatne jest uruchomienie jakiegoś polecenia na hoście zdalnym, ale z możliwością wprowadzania danych i oglądania wyniku przez sieć.

Tradycyjne polecenia używane do wykonywania poleceń na hostach zdalnych to *rlogin*, *rsh* i *rcp*. Przykład polecenia *rlogin* widzieliśmy w rozdziale 1, *Wprowadzenie do sieci*. Tamże, w podrozdziale *Bezpieczeństwo systemu* krótko omówiliśmy zagadnie-

nia bezpieczeństwa związane z tym poleceniem i zasugerowaliśmy zastąpienie go przez *ssh*. Pakiet *ssh* zawiera zamienniki w postaci *slogin*, *ssh* i *scp*.

Każde z tych poleceń uruchamia powłokę na zdalnym hoście i pozwala użytkownikowi na wykonywanie poleceń. Oczywiście klient musi mieć konto na hoście zdalnym, na którym jest wykonywane polecenie. Tak więc, wszystkie polecenia wykorzystują proces uwierzytelniania. Polecenia *r* używają po prostu nazwy użytkownika i hasła, które wymieniają pomiędzy hostami bez szyfrowania, a więc każdy, kto słucha może łatwo przejąć hasła. Pakiet poleceń *ssh* zapewnia wyższy poziom bezpieczeństwa, gdyż wykorzystuje technikę *kodowania informacji z kluczem wielodostępu* (ang. *Public Key Cryptography*), która zapewnia uwierzytelnianie i szyfrowanie pomiędzy hostami, co z kolei daje pewność, że ani hasła, ani dane sesji nie zostaną łatwo przechwycone przez inne hosty.

Pewnym użytkownikom można ułatwić sprawdzanie haseł. Na przykład, jeżeli często logujesz się na komputerach w swojej sieci LAN, możesz być wpuszczany bez potrzeby ciągłego wpisywania hasła. Polecenia typu *r* także dawały taką możliwość, ale pakiet *ssh* pozwala to zrobić nieco prościej. Zaznaczmy, że nie jest to żadna rewelacja. Po prostu, jeżeli zdobędzie się już konto na jednej maszynie, można uzyskać dostęp do wszystkich pozostałych kont, które użytkownik skonfigurował do logowania się bez hasła. Jest to dosyć wygodne i dla tego użytkownicy często sięgają po to rozwiązanie.

Omówmy usuwanie poleceń *r* i uruchamianie zamiast nich pakietu *ssh*.

Wyłączanie poleceń r

Rozpocznijmy od usuwania poleceń *r*. Najprostszym sposobem na wyłączenie poleceń *r* jest poprzedzenie komentarzem (lub usunięcie) wpisów w pliku */etc/inetd.conf*. Interesujące nas wpisy wyglądają następująco:

```
# Shell, login, exec i talk to protokoły BSD
shell    stream  tcp   nowait  root   /usr/sbin/tcpd /usr/sbin/in.rshd
login    stream  tcp   nowait  root   /usr/sbin/tcpd /usr/sbin/in.rlogind
exec     stream  tcp   nowait  root   /usr/sbin/tcpd /usr/sbin/in.rexecd
```

Możesz je zakomentować, umieszczając znak # na początku każdego wiersza, lub zupełnie usunąć wiersze. Pamiętaj, że musisz uruchomić ponownie demona *inetd*, aby zmiana odniosła skutek. Najlepiej byłoby usunąć również same programy demonów.

Instalowanie i konfigurowanie ssh

OpenSSH jest darmową wersją pakietu programów *ssh*. Wersję dla Linuksa można znaleźć pod adresem *http://violet.ibs.com.au/openssh/* i w większości najnowszych dystrybucji*. Nie będziemy opisywali tu taj kompilacji. Dobre instrukcje znajdują się w pakiecie źródłowym. Jeżeli możesz zainstalować już skompilowany pakiet, warto to zrobić.

* OpenSSH został stworzony w ramach projektu OpenBSD i jest doskonałym przykładem korzyści z darmowego oprogramowania.

Sesja *ssh* angażuje dwie strony. Jedna z nich to klient *ssh*, którego musisz skonfigurować i uruchomić na hoście lokalnym, a druga to demon *ssh*, który musi działać na hoście zdalnym.

Demon ssh

Demon *sshd* to program, który oczekuje połączeń sieciowych od klientów *ssh*, obsługuje uwierzytelnienie i wykonuje żądane polecenia. Ma on jeden plik konfiguracyjny o nazwie */etc/ssh/sshd_config* i specjalny, reprezentujący hosta plik, który zawiera klucz używany w procesach uwierzytelniania i szyfrowania. Każdy host i każdy klient mają własny klucz.

W pakiecie umieszczane jest też narzędzie *ssh-keygen*, które służy do generowania klucza losowego. Zwykle jest używane w czasie instalacji do wygenerowania klucza hosta, który administrator zwykle umieszcza w pliku */etc/ssh/ssh_host_key*. Klucze mogą mieć dowolną długość większą od 512 bitów. Domyślnie *ssh_keygen* generuje klucze długości 1024 bitów, a większość osób tę wartość domyślną uznaje. Aby wygenerować klucz losowy, musisz wywołać polecenie *ssh-keygen* w następujący sposób:

```
# ssh-keygen -f /etc/ssh/ssh_host_key
```

Zostaniesz poproszony o wprowadzenie frazy (ang. *passphrase*). Jednak klucze hosta nie muszą wykorzystywać frazy, a więc po prostu naciśnij [Enter] i przejdź dalej. Wynik działania programu będzie następujący:

```
Generating RSA keys: ......oooooO..............................oooooO
Key generation complete.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /etc/ssh/ssh_host_key
Your public key has been saved in /etc/ssh/ssh_host_key.pub
The key fingerprint is:
1024 3a:14:78:8e:5a:a3:6b:bc:b0:69:10:23:b7:d8:56:82 root@moria
```

Widać, że zostały utworzone dwa pliki. Pierwszy, */etc/ssh/ssh_host_key*, jest nazywany kluczem prywatnym i musi być trzymany w tajemnicy. Drugi, */etc/ssh/ssh_host_key.pub*, jest nazywany kluczem publicznym i ten możesz udostępniać.

Kiedy masz klucze *ssh* do komunikacji, musisz stworzyć plik konfiguracyjny. Pakiet *ssh* ma wiele możliwości i plik konfiguracyjny może zawierać wiele opcji. Pokażemy prosty przykład, od którego łatwo ci będzie zacząć. Aby włączyć dalsze funkcje, powinieneś zajrzeć do dokumentacji. Poniższy kod pokazuje bezpieczny i minimalny plik konfiguracyjny *sshd*. Pozostałe opcje konfiguracyjne są szczegółowo opisane na stronie podręcznika elektronicznego *sshd*(8):

```
# /etc/ssh/sshd_config
#

# Adresy IP, na których oczekiwane s□ po□□czenia. 0.0.0.0
# oznacza wszystkie adresy lokalne.
ListenAddress 0.0.0.0

# Port TCP, na którym oczekiwane s□ po□□czenia. Domy□lnie 22.
Port 22
```

```
# Nazwa pliku z kluczem hosta.
HostKey /etc/ssh/ssh_host_key

# D□ugo□□ klucza w bitach.
ServerKeyBits 1024

# Czy pozwalamy na logowanie roota przez ssh?
PermitRootLogin no

# Czy demon ssh powinien sprawdza□ katalogi macierzyste
# u□ytkowników i prawa dost□pu do plików przed pozwoleniem na
# zalogowanie?
StrictModes yes

# Czy pozwalamy na metod□ uwierzytelniania przez ~/.rhosts i
# /etc/hosts.equiv
RhostsAuthentication no
# Czy pozwalamy na czyste uwierzytelnienie RSA?
RSAAuthentication yes
# Czy pozwalamy na uwierzytelnienie has□em?
PasswordAuthentication yes

# Czy pozwalamy na uwierzytelnienie RSA w po□□czeniu z
# /etc/hosts.equiv?
RhostRSAAuthentication no
# Czy powinni□my ignorowa□ pliki ~/.rhosts?
IgnoreRhosts yes

# Czy pozwalamy na logowanie na konta bez hase□?
PermitEmptyPasswords no
```

Ważne jest sprawdzenie praw dostępu do plików konfiguracyjnych, gdyż ich poprawność ma wpływ na bezpieczeństwo systemu. Użyj poniższych poleceń:

```
# chown -R root:root /etc/ssh
# chmod 755 /etc/ssh
# chmod 600 /etc/ssh/ssh_host_key
# chmod 644 /etc/ssh/ssh_host_key.pub
# chmod 644 /etc/ssh/sshd_config
```

Ostatnim etapem administracji *sshd* jest uruchomienie demona. Zwykle tworzysz dla niego plik *rc* lub dodajesz go do istniejącego, aby był automatycznie uruchamiany przy starcie komputera. Demon działa samodzielnie i nie potrzebuje żadnego wpisu w pliku */etc/inetd.conf*. Demon musi być uruchomiony jako użytkownik root. Składnia jest bardzo prosta:

```
/usr/sbin/sshd
```

Demon *sshd* automatycznie przejdzie w tło zaraz po uruchomieniu. Od tej chwili jesteś gotów przyjmować połączenia *ssh*.

Klient ssh

Istnieje kilka programów klientów *ssh*: *slogin*, *scp* i *ssh*. Każdy z nich odczytuje ten sam plik konfiguracyjny, zwykle */etc/ssh/ssh_config*. Każdy czyta także pliki konfiguracyjne z podkatalogu *.ssh* katalogu macierzystego użytkownika uruchamiającego klienta. Najważniejsze z tych plików to *.ssh/config*, który może zawierać opcje o wyższym priorytecie niż te w pliku */etc/ssh/ssh_config*, plik *.ssh/identity*, który zawiera

prywatny klucz użytkownika, oraz odpowiedni plik *.ssh/identity.pub* zawierający publiczny klucz użytkownika. Inne ważne pliki to *.ssh/known_hosts* i *.ssh/authorized_keys*. Omówimy je dalej w podrozdziale *Korzystanie z ssh*. Najpierw przygotujmy globalny plik konfiguracyjny i plik klucza użytkownika.

Plik */etc/ssh/ssh_config* jest bardzo podobny do pliku konfiguracyjnego serwera. Analogicznie, zawiera wiele funkcji, które można konfigurować, ale minimalna konfiguracja wygląda tak, jak pokazano w przykładzie 12-5. Pozostałe opcje konfiguracyjne są szczegółowo omówione na stronach podręcznika elektronicznego *sshd(8)*. Możesz dodać części odpowiedzialne za określone hosty lub grupy hostów. Parametrem dyrektywy „`Host`" może być zarówno pełna nazwa hosta, jak i nazwa zapisana za pomocą znaków uniwersalnych (ang. *wildcards*), czego użyliśmy w przykładzie, by uwzględnić wszystkie hosty. Moglibyśmy stworzyć na przykład wpis, który używałby `Host *.vbrew.com`; wtedy pasowałyby do niego wszystkie hosty z domeny `vbrew.com`.

Przykład 12-5. Przykładowy plik konfiguracyjny klienta ssh

```
# /etc/ssh/ssh_config

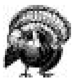
# Domy□lne opcje u□ywane przy po□□czeniu z hostem zdalnym
Host *
  # Kompresowa□ dane w czasie sesji?
  Compression yes
  # .. u□ywaj□c którego poziomu kompresji? (1 - szybka/s□aba, 9 - wolna/dobra)
  CompressionLevel 6

  # Czy skorzysta□ z rsh, je□eli po□□czenie bezpieczne si□ nie uda?
  FallBackToRsh no

  # Czy powinni□my wysy□a□ komunikaty o aktywno□ci?
  # Przydatne, je□eli korzystasz z maskowania IP
  KeepAlive yes

  # Próbowa□ uwierzytelniania RSA?
  RSAAuthentication yes
  # Próbowa□ uwierzytelniania RSA w po□□czeniu z
  # uwierzytelnianiem .rhosts?
  RhostsRSAAuthentication yes
```

W podrozdziale dotyczącym konfiguracji serwera wspomnieliśmy, że każdy host i użytkownik mają klucz. Klucz użytkownika jest umieszczony w pliku *~/.ssh/identity*. Aby wygenerować klucz, posługujesz się tym samym poleceniem *ssh-keygen*, które służyło do generowania klucza hosta, tylko, że tym razem nie potrzebujesz podawać nazwy pliku, w którym zapisujesz klucz. *ssh-keygen* domyślnie umieszcza go w odpowiednim miejscu, ale pyta cię o nazwę pliku na wypadek, gdybyś chciał go zapisać gdzie indziej. Czasem warto mieć kilka kluczy tożsamości, a więc *ssh* na to pozwala. Tak jak przedtem, *ssh-keygen* pyta cię o wprowadzenie frazy. Frazy dają dodatkowy poziom bezpieczeństwa i są dobrym rozwiązaniem. Twoja fraza nie będzie wyświetlana na ekranie podczas wprowadzania.



Nie ma możliwości odtworzenia frazy, gdybyś jej zapomniał. Wymyśl więc coś, co zapamiętasz, ale tak jak w przypadku wszystkich haseł, niech to nie będzie coś oczywistego, pospolity rzeczownik ani twoje imię. Aby fraza była naprawdę skuteczna, powinna liczyć od 10 do 30 znaków i nie może być zwykłym wyrażeniem. Spróbuj wtrącić kilka nietypowych znaków. Jeżeli zapomnisz frazy, będziesz musiał wygenerować nowy klucz.

Powinieneś poprosić wszystkich swoich użytkowników o uruchomienie polecenia *ssh-keygen*, by mieć pewność, że ich klucz został stworzony poprawnie. *ssh-keygen* utworzy za nich katalogi *~/.ssh/* z odpowiednimi prawami dostępu oraz stworzy klucze prywatny i publiczny odpowiednio w plikach *.ssh/identity* i *.ssh/identity.pub*. Przykładowa sesja powinna wyglądać tak:

```
$ ssh-keygen
Generating RSA keys: .......ooooooO...............................
Key generation complete.
Enter file in which to save the key (/home/maggie/.ssh/identity):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/maggie/.ssh/identity.
Your public key has been saved in /home/maggie/.ssh/identity.pub.
The key fingerprint is:
1024 85:49:53:f4:8a:d6:d9:05:d0:1f:23:c4:d7:2a:11:67 maggie@moria
$
```

Teraz *ssh* jest gotowe do pracy.

Korzystanie z ssh

Mamy już zainstalowane polecenie *ssh* wraz z towarzyszącymi mu narzędziami pomocniczymi. Wszystko jest gotowe do pracy. Przyjrzyjmy się teraz, jak można je uruchamiać.

Najpierw spróbujemy zalogować się do zdalnego hosta. Możemy posłużyć się programem *slogin* w prawie identyczny sposób, jak używaliśmy programu *rlogin* we wcześniejszym przykładzie w tej książce. Gdy za pierwszym razem próbujesz się łączyć z hostem, klient *ssh* odczytuje klucz publiczny hosta i pyta cię, czy potwierdzasz jego tożsamość, pokazując skróconą wersję klucza publicznego nazywaną *odciskiem palca* (ang. *fingerprint*).

Administrator hosta zdalnego powinien wcześniej dostarczyć ci „odcisk palca” klucza publicznego tego hosta. Ten klucz powinieneś dodać do swojego pliku *.ssh/known_hosts*. Jeżeli administrator zdalnego hosta tego nie zrobił, możesz połączyć się z hostem zdalnym, ale *ssh* ostrzeże cię, że nie ma klucza i zapyta, czy chcesz przyjąć ten oferowany przez host zdalny. Zakładając, że jesteś pewien, że nikt nie podszywa się pod DNS i że połączyłeś się z poprawnym hostem, możesz wybrać odpowiedź *yes*. Odpowiedni klucz zostanie zapisany automatycznie w twoim pliku *.ssh/known_hosts* i nie będziesz o niego pytany ponownie. Jeżeli przy następnej próbie połączenia klucz publiczny uzyskany z danego hosta nie będzie pasował do tego, który przechowujesz w pliku *.ssh/known_hosts*, otrzymasz ostrzeżenie, ponieważ naruszato bezpieczeństwo.

Pierwsze logowanie do zdalnego hosta będzie wyglądało jakoś tak:

```
$ slogin vchianti.vbrew.com
The authenticity of host 'vchianti.vbrew.com' can't be established.
Key fingerprint is 1024 7b:d4:a8:28:c5:19:52:53:3a:fe:8d:95:dd:14:93:f5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permamently added 'vchianti.vbrew.com,172.16.2.3' to the list of/
    known hosts.
maggie@vchianti.vbrew.com's password:
Last login: Thu Feb  1 23:28:58 2000 from vstout.vbrew.com
$
```

Zostaniesz poproszony o podanie hasła, które powinieneś wprowadzić na konto na hoście zdalnym, a nie lokalnym. Hasło to nie pojawia się na ekranie w czasie wpisywania.

Jeżeli nie podamy żadnych szczególnych argumentów, *slogin* spróbuje zalogować się z tym samym identyfikatorem użytkownika, jaki jest używany na komputerze lokalnym. Możesz to zmienić, używając argumentu `-l`, w którym podaje się inną nazwę użytkownika logującego się do hosta zdalnego. Tak właśnie robiliśmy we wcześniejszym przykładzie.

Za pomocą programu *scp* możemy kopiować pliki zarówno ze zdalnego hosta, jak i do niego. Składnia jest podobna do typowego *cp* z tą różnicą, że musisz podać nazwę hosta przed nazwą pliku, co oznacza, że ścieżka do tyczy podanego hosta. Poniższy przykład, w którym lokalny plik */tmp/fred* jest kopiowany do katalogu */home/maggie* na hoście **chianti.vbrew.com**, pokazuje składnię *scp*:

```
$ scp /tmp/fred vchianti.vbrew.com:/home/maggie/
maggie@vchianti.vbrew.com's password:
fred                 100% |*****************************| 50165 00:01 ETA
```

Znów zostaniesz poproszony o wprowadzenie hasła. Polecenie *scp* domyślnie wyświetla przydatne komunikaty informujące o postępie. Równie łatwo możesz skopiować plik z hosta zdalnego. Po prostu podaj nazwę hosta i ścieżkę jako plik źródłowy, a ścieżkę lokalną jako plik docelowy. Możliwe jest również skopiowanie pliku z hosta zdalnego na inny host zdalny, ale zwykle się tego nie robi, ponieważ wszystkie dane przechodzą przez twój host.

Za pomocą *ssh* możesz uruchamiać polecenia na hoście zdalnym. Znów składnia jest bardzo prosta. Niech użytkownik `maggie` obejrzy zawartość katalogu głównego hosta **vchianti.vbrew.com**. Może to zrobić następująco:

```
$ ssh vchianti.vbrew.com ls -CF /
maggie@vchianti.vbrew.com's password:
bin/    console@  dos/     home/    lost+found/  pub@   tmp/  vmlinuz@
boot/   dev/      etc/     initrd/  mnt/         root/  usr/  vmlinuz.old@
cdrom/  disk/     floppy/  lib/     proc/        sbin/  var/
```

Polecenie *ssh* możesz umieścić w potoku poleceń i przekazywać wejście/wyjście programu do lub z innego programu, tak jak w innych poleceniach, z tą różnicą, że wejście lub wyjście są kierowane do lub z hosta zdalnego przez połączenie *ssh*. Oto przykład, jak możesz wykorzystać tę możliwość w połączeniu z poleceniem *tar* do przekopiowania całego katalogu (z podkatalogami i plikami) z hosta zdalnego na host lokalny:

```
$ ssh vchianti.vbrew.com "tar cf - /etc/" | tar xvf -
maggie@vchianti.vbrew.com's password:
etc/GNUstep
etc/Muttrc
etc/Net
etc/X11
etc/adduser.conf
..
..
```

Polecenie do wykonania wzięliśmy tutaj w cudzysłów, aby było jasne, co przekazujemy jako argument do *ssh* i co jest używane przez lokalną powłokę. Polecenie to wykonuje *tar* na hoście zdalnym, które z kolei archiwizuje katalog */etc* i wypisuje wynik na standardowe wyjście. Zastosowaliśmy potok, przez który przekazujemy standardowe wyjście do polecenia *tar* działającego na hoście lokalnym w trybie odczytywania ze standardowego wejścia.

Znów zostaliśmy poproszeni o wprowadzenie hasła. Teraz możesz zobaczyć, dlaczego zachęcaliśmy cię do skonfigurowania *ssh* tak, żeby nie pytało o hasła za każdym razem! Skonfigurujmy teraz naszego lokalnego klienta *ssh* tak, by nie pytał o hasło przy łączeniu się z hostem **vchianti.vbrew.com**. Wspomnieliśmy wcześniej o pliku *.ssh/authorized_keys*. Właśnie on jest używany do tego celu. Plik *.ssh/authorized_keys* zawiera klucze *publiczne* wszelkich zdalnych kont użytkowników, na które chcemy się automatycznie logować. Automatyczne logowanie możesz ustawić, kopiując zawartość pliku *.ssh/identity.pub* z konta *zdalnego* do lokalnego pliku *.ssh/authorized_keys*. Istotne jest, by prawa dostępu do pliku *.ssh/authorized_keys* pozwalały na dostęp tylko tobie i tylko na zapis i odczyt. W przeciwnym razie ktoś mógłby ukraść klucze i zalogować się na zdalne konto. Aby mieć pewność, że prawa dostępu są poprawne, zmień *.ssh/authorized_keys* w następujący sposób:

```
$ chmod 600 ~/.ssh/authorized_keys
```

Klucze publiczne to jeden długi wiersz czystego tekstu. Jeżeli używasz funkcji kopiowania i wklejania do powielania kluczy w twoim pliku lokalnym, pamiętaj o usunięciu wszelkich znaków końca wiersza, które mogły zostać przez przypadek wstawione. Plik *.ssh/authorized_keys* może zawierać wiele takich kluczy, ale każdy z nich musi znajdować się w oddzielnym wierszu.

Pakiet narzędzi *ssh* oferuje wiele przydatnych funkcji i opcji, które warto poznać. W poszukiwaniu dodatkowych informacji zajrzyj do podręcznika elektronicznego i dokumentacji dostarczanej wraz z pakietem.

13



System informacji sieciowej

Gdy obsługujesz sieć lokalną, zwykle twoim głównym celem jest zapewnienie swoim użytkownikom takiego środowiska, w którym sieć jest przezroczysta. Warunkiem jest synchronizacja na wszystkich hostach w sieci takich danych, jak informacje o kontach użytkowników. Wówczas użytkownicy mogą swobodnie przesiadać się z komputera na komputer bez potrzeby pamiętania różnych haseł i kopiowania danych między maszynami. Dane, które są przechowywane centralnie, nie muszą być replikowane, jeśli istnieje jakiś wygodny sposób na dostanie się do nich z hosta podłączonego do sieci. Centralne przechowywanie istotnych informacji administracyjnych ma szereg zalet. Gwarantuje spójność danych. Daje większą swobodę użytkownikom poprzez możliwość przesiadania się z hosta do hosta. Ułatwia życie administratorowi systemu, który zarządza tylko jednym „egzemplarzem” informacji.

Wcześniej omówiliśmy ważny przykład centralizacji danych, pochodzący z Internetu – system nazw domen (DNS). DNS udostępnia ograniczony zakres informacji, z których najważniejsze to tłumaczenie nazw hostów na adresy IP i odwrotnie. Inne typy danych nie mają swoich specjalistycznych usług. Co więcej, jeżeli zarządzasz tylko małą siecią LAN bez dostępu do Internetu, korzyści ze skonfigurowania DNS-u mogą nie być warte pracy, jaką trzeba w to włożyć.

Dlatego właśnie firma Sun stworzyła *system informacji sieciowej* (*Network Information System* – NIS). NIS to funkcje ogólnego dostępu do bazy danych. Za ich pomocą można dystrybuować do wszystkich hostów w sieci na przykład informacje zawarte w plikach *passwd* i *group*. Dzięki temu sieć jest widoczna jako jeden system, z tymi samymi kontami na wszystkich hostach. Podobnie możesz wykorzystać NIS-a do dystrybuowania informacji o nazwie hosta zawartych w pliku */etc/hosts* do wszystkich innych maszyn w sieci.

NIS jest oparty na RPC i składa się z serwera, biblioteki strony klienta i kilku narzędzi administracyjnych. Pierwotnie nosił nazwę *Yellow Pages* (lub YP); nadal można

spotkać się z odwołaniami do niej. Jednak okazało się, że nazwa ta jest znakiem towarowym firmy British Telecom, która zażądała, by Sun przestał jej używać. Jak wiadomo, niektóre nazwy łatwo się zapamiętuje i dlatego „YP" pozostało jako przedrostek w większości poleceń związanych z NIS-em, jak *ypserv* i *ypbind*.

Obecnie NIS jest dostępny praktycznie we wszystkich Uniksach i istnieją nawet darmowe jego implementacje. BSD Net-2 zawiera implementację, która pochodzi od wzorcowej implementacji publicdomain finansowanej przez Suna. Kod biblioteki klienta z tej wersji znajdował się przez długi czas w linuksowej bibliotece *libc*, a programy administracyjne zostały przeniesione do Linuksa przez Swena Thümmlera*. Jednak we wzorcowej implementacji brakuje serwera NIS.

Peter Eriksson przygotował nową implementację o nazwie NYS**. Obsługuje ona zarówno NIS-a, jak i jego rozszerzoną wersję NIS+. NYS nie tylko udostępnia zestaw narzędzi i serwer NIS-a, ale także cały nowy zestaw funkcji bibliotecznych, które trzeba wkompilować w bibliotekę *libc*, jeżeli chcesz ich używać. Należy do nich nowy sposób konfiguracji rozwiązywania nazw, który zastępuje aktualny schemat oparty na pliku *host.conf*.

Biblioteka GNU *libc*, znana jako *libc6* w społeczności Linuksa, zawiera uaktualnioną wersję tradycyjnego NIS-a autorstwa Thorstena Kukuka***. Obsługuje ona wszystkie funkcje biblioteczne udostępnione przez NYS-a i wykorzystuje również rozszerzony schemat konfiguracji NYS. Wciąż potrzebne ci będą narzędzia i serwer, ale użycie biblioteki GNU *libc* zaoszczędzi ci problemów z poprawianiem i kompilowaniem biblioteki.

Ten rozdział jest poświęcony procedurom obsługi NIS-a zawartym w bibliotece GNU *libc*, a nie tej z dwóch pozostałych pakietów. Gdybyś chciał uruchomić któryś z tych pakietów, instrukcje zawarte w tym rozdziale mogą nie wystarczyć. Dodatkowych informacji szukaj w dokumencie *NIS-HOWTO* lub w książce *Managing NFS and NIS* autorstwa Hala Sterna (wyd. O'Reilly).

Poznawanie NIS-a

Swoją bazę z informacjami NIS przechowuje w plikach zwanych mapami (ang. *maps*), które z kolei zawierają pary klucz-wartość. Przykład pary klucz-wartość to nazwa użytkownika i zaszyfrowana postać jego hasła. Plik jest przechowywany na centralnym hoście, na którym działa serwer NIS-a i z którego klienty mogą pobierać informacje, używając różnych wywołań RPC. Często mapy są przechowywane w plikach DBM****.

---

* Ze Swenem można się skontaktować pod adresem *swen@uni-paderborn.de*. Klienty NIS-a są dostępne w pakiecie *yp-linux.tar.gz* pod adresem **metalab.unc.edu** w katalogu *system/Network*.

** Z Peterem można się skontaktować pod adresem *pen@lysator.liu.se*. Obecna wersja NYS to 1.2.8.

*** Z Thorstenem można się skontaktować pod adresem *kukuk@uni-paderborn.de*).

**** DBM to prosta biblioteka do zarządzania bazami danych, wykorzystująca techniki mieszające do przyśpieszenia operacji wyszukiwania. Istnieje darmowa implementacja DBM stworzona w ramach projektu GNU. Nosi ona nazwę *gdbm* i jest częścią większości dystrybucji Linuksa.

Same mapy są zwykle generowane na podstawie głównych plików tekstowych, takich jak */etc/hosts* czy */etc/passwd*. Niektóre pliki potrzebują po kilka map, po jednej dla każdego typu klucza poszukiwań. Na przykład w pliku *hosts* możesz poszukiwać nazwy hosta lub adresu IP. W związku z tym na jego podstawie są tworzone dwie mapy NIS o nazwach *hosts.byname* i *hosts.byaddr*. Tabela 13-1 pokazuje powszechnie używane mapy i pliki, na podstawie których są one tworzone.

Tabela 13-1. Niektóre standardowe mapy NIS-a i odpowiadające im pliki

Plik główny	*Map(y)*	*Opis*
/etc/hosts	*hosts.byname*, *hosts.byaddr*	Odwzorowuje adresy IP na nazwy hostów.
/etc/networks	*networks.byname*, *networks.byaddr*	Odwzorowuje adresy sieci na ich nazwy.
/etc/passwd	*passwd.byname*, *passwd.byuid*	Odwzorowuje zaszyfrowane hasła na nazwy użytkowników.
/etc/group	*group.byname*, *group.bygid*	Odwzorowuje ID grupy na jej nazwę.
/etc/services	*services.byname*, *services.bynumber*	Odwzorowuje opisy usług na ich nazwy.
/etc/rpc	*rpc.byname*, *rpc.bynumber*	Odwzorowuje numery usług Sun RPC na nazwy usług RPC.
/etc/protocols	*protocols.byname*, *protocols.bynumber*	Odwzorowuje numery protokołów na ich nazwy.
/usr/lib/aliases	*mail.aliases*	Odwzorowuje aliasy pocztowe na nazwy aliasów pocztowych.

Pakiety NIS obsługują także inne pliki i mapy. Zawierają one zwykle informacje dla aplikacji nie omawianych w tej książce, taką jest mapa *bootparams* używana przez serwer Suna *bootparamd*.

Dla niektórych map powszechnie używa się *skrótów*, które są łatwiejsze do wpisywania. Należy jednak pamiętać, że jedynie *ypcat* i *ypmatch* – dwa narzędzia sprawdzające konfigurację NIS-a – potrafią rozwiązywać te skróty. Aby uzyskać pełną listę skrótów interpretowanych przez te narzędzia, uruchom następujące polecenie:

```
$ ypcat -x
Use "passwd" for "passwd.byname"
Use "group" for "group.byname"
Use "networks" for "networks.byaddr"
Use "hosts" for "hosts.byaddr"
Use "protocols" for "protocols.bynumber"
Use "services" for "services.byname"
Use "aliases" for "mail.aliases"
Use "ethers" for "ethers.byname"
```

Program serwera NIS jest tradycyjnie nazywany *ypserv*. Pojedynczy serwer zwykle wystarcza dla sieci średnich rozmiarów. W dużych sieciach możesz zdecydować się na uruchomienie kilku serwerów na różnych komputerach i w różnych segmentach sieci, aby rozłożyć obciążenie pomiędzy serwerami i ruterami. Serwery są synchro-

nizowane przez wybranie jednego *serwera głównego* (ang. *master server*) i ustawienie pozostałych serwerów jako *podrzędnych* (ang. *slave servers*). Mapy są tworzone tylko na serwerze głównym. Z niego są dystrybuowane do serwerów podrzędnych.

Dosyć skrótowo omówiliśmy pojęcie „sieci". W NIS-ie istnieje ważny termin na określenie zbioru wszystkich hostów, które mają wspólną konfigurację rozpowszechnianą przez NIS: *domena NIS*. Jednak domeny NIS nie mają nic wspólnego z domenami, które spotkaliśmy przy omawianiu DNS-u. Aby uniknąć dwuznaczności, w tym rozdziale zawsze będziemy wskazywać typ domeny, o której mówimy.

Domeny NIS pełnią funkcję czysto administracyjną. Są w zasadzie niewidoczne dla użytkowników, z wyjątkiem dzielenia haseł pomiędzy wszystkimi maszynami w domenie. Dlatego nazwa nadana domenie NIS jest istotna tylko dla administratorów. Zwykle może to być dowolna nazwa: chodzi tylko o to, aby była inna niż nazwy pozostałych domen NIS w twojej sieci lokalnej. Na przykład administrator browaru wirtualnego może stworzyć dwie domeny NIS, jedną dla samego browaru, a drugą dla winiarni. Mógłby nadać im odpowiednio nazwy **brewery** i **winery**. Innym powszechnie stosowanym schematem jest po prostu używanie nazw domenowych DNS także dla NIS-a.

Aby nadać nazwę domenie NIS, do której należy twój host, i ją wyświetlić, możesz użyć polecenia *domainname*. Wywołanie bez argumentów powoduje wypisanie aktualnej nazwy domeny NIS. Aby nadać nazwę domenie, musisz uzyskać prawa użytkownika uprzywilejowanego:

```
# domainname brewery
```

Domeny NIS określają, do którego serwera NIS będą wysyłane zapytania. Na przykład program *login* na hoście winiarni powinien oczywiście wysyłać zapytania o hasło użytkownika tylko do serwera NIS winiarni (lub jednego z nich, jeżeli jest ich kilka), natomiast aplikacja działająca na hoście browaru powinna trzymać się serwera należącego do browaru.

Pozostała jeszcze jedna zagadka do wyjaśnienia: skąd klient wie, z którym serwerem ma się połączyć? Najprostsze rozwiązanie to użycie pliku konfiguracyjnego, w którym znajduje się nazwa hosta, na którym działa serwer. Jednak podejście to jest mało elastyczne, ponieważ nie pozwala klientom na używanie różnych serwerów (z tej samej domeny oczywiście) w zależności od ich dostępności. Dlatego implementacje NIS-a opierają się na specjalnym demonie *ypbind*, dzięki któremu można wykryć odpowiedni serwer NIS w danej domenie. Przed wysłaniem zapytań NIS, aplikacja najpierw ustala za pomocą *ypbind*, którego serwera ma używać.

ypbind znajduje serwery przez rozgłaszanie zapytania w lokalnej sieci IP. Pierwszy serwer, który odpowie, jest uznawany za najszybszy i wykorzystywany we wszystkich kolejnych zapytaniach NIS. Po pewnym czasie *ypbind* ponownie szuka aktywnych serwerów. Robi tak również, jeżeli serwer nie jest dostępny.

Dynamiczne przypisywanie jest przydatne tylko wtedy, gdy twoja sieć udostępnia więcej niż jeden serwer NIS. Trzeba jednak pamiętać, że zagraża bezpieczeństwu. *ypbind* na śle po wierzy temu, kto odpowie, bez względu na to, czy jest to oficjalny serwer NIS, czy złośliwy intruz. Nie trzeba mówić, że jest to szczególnie niebez-

pieczne, jeżeli za pomocą NIS-a zarządzasz bazami danych haseł. Aby uchronić się przed kłopotami, linuksowy program *ypbind* posiada opcję szukania serwera NIS w sieci lokalnej lub podania nazwy hosta, na którym działa serwer NIS, w pliku konfiguracyjnym.

NIS kontra NIS+

Cel i nazwa to jedyne rzeczy, które łączą NIS-a i NIS+. NIS+ jest zbudowany zupełnie inaczej niż NIS. Zamiast płaskiej przestrzeni nazw z chaotycznymi domenami NIS-a, NIS+ wykorzystuje hierarchiczną strukturę nazw podobną do DNS-u. Zamiast map, używane są tak zwane *tabele*, które składają się z wierszy i kolumn. Każdy wiersz zawiera obiekt z bazy danych NIS+, a każda kolumna opisuje własności obiektu NIS+. Każda tabela dla danej domeny NIS+ zawiera dane z domeny nadrzędnej. Ponadto wpis w tabeli może zawierać dowiązanie do innej tabeli. Te funkcje umożliwiają porządkowanie informacji na wiele sposobów.

NIS+ dodatkowo obsługuje bezpieczne i szyfrowane RPC, co znacznie pomaga w rozwiązywaniu problemów z bezpieczeństwem, które ujawniły się w przypadku NIS-a.

Tradycyjny NIS używa RPC w wersji 2, natomiast NIS+ – w wersji 3. W czasie pisania tej książki nie ma jeszcze dobrze działającej implementacji NIS+ dla Linuksa, dlatego nie poświęcamy więcej miejsca temu tematowi.

NIS – strona klienta

Jeżeli znasz się na pisaniu lub przenoszeniu aplikacji sieciowych, zapewne zauważyłeś, że większość przedstawionych map NIS odpowiada funkcjom bibliotecznym z biblioteki C. Na przykład, aby uzyskać informacje o *passwd*, używasz funkcji *getpwnam* i *getpwuid*, zwracających informacje o koncie związane odpowiednio z daną nazwą użytkownika lub jego ID. W normalnych warunkach funkcje te realizują przeszukiwanie standardowego pliku, na przykład */etc/passwd*.

Implementacja tych funkcji uwzględniająca NIS-a modyfikuje jednak to zachowanie przez dodanie do serwera NIS wywołania RPC, które szuka nazwy użytkownika lub jego ID. Dzieje się to niewidocznie dla aplikacji. Funkcja może traktować dane NIS tak, jak by były zawarte w oryginalnym pliku *passwd*, a więc oba zestawy informacji są dostępne dla aplikacji i przez nią wykorzystywane, lub tak, jak by zupełnie zastępowały dane w *passwd* i wtedy je ignoruje, a wykorzystuje tylko dane NIS.

W tradycyjnych implementacjach NIS-a obowiązywały pewne ustalenia co do tego, kiedy mapy były zastępowane innymi, a kiedy dodawane do oryginalnych informacji. Niektóre mapy, takie jak *passwd*, wymagały modyfikacji pliku *passwd*. Jeżeli została ona wykonana niepoprawnie, tworzyły się dziury w bezpieczeństwie. Aby uniknąć tych pułapek, NYS i GNU *libc* wykorzystują ogólny schemat konfiguracji określający, czy dany zestaw funkcji klienta używa oryginalnych plików, plików NIS czy NIS+ i w jakiej kolejności. Ten schemat będzie opisany w dalszej części tego rozdziału.

Eksploatowanie serwera NIS

Dość już tej teoretycznej paplaniny. Czas przyjrzeć się, jak działa rzeczywista konfiguracja. W tym podrozdziale omówimy konfigurację serwera NIS. Jeżeli serwer NIS już działa w twojej sieci, nie musisz tworzyć własnego i możesz bez szkody dla siebie pominąć ten podrozdział.

Pamiętaj, że jeżeli zamierzasz eksperymentować z serwerem, musisz uważać, by nie zdublować nazwy domeny NIS. Mogłoby dojść do awarii usługi w całej sieci. Nie mówiąc już o zdenerwowaniu współużytkowników.

Istnieją dwie możliwe konfiguracje serwera NIS: jako serwera głównego i jako podrzędnego. Konfiguracja serwera podrzędnego daje działający komputer zapasowy na wypadek, gdyby serwer główny uległ awarii. Omówimy tu tajedynie konfigurację serwera głównego. Dokumentacja serwera wyjaśnia różnice między tymi dwoma konfiguracjami, a więc zajrzyj do niej, gdybyś musiał skonfigurować serwer podrzędny.

Obecnie istnieją dwa darmowe serwery NIS dostępne dla Linuksa: jeden zawarty w pakiecie *yps* Tobiasa Rebera i drugi – w pakiecie *ypserv* Petera Erikssona. Nie gra roli, który z nich uruchomisz.

Po zainstalowaniu programu serwera (*ypserv*) w katalogu */usr/sbin*, powinieneś stworzyć katalog, w którym będą przechowywane pliki map rozpowszechniane przez twój serwer. Przy konfigurowaniu domeny NIS dla domeny **brewery**, mapy będą umieszczone w katalogu */var/yp/brewery*. Za nim serwer zdecyduje się obsługiwać konkretną domenę, sprawdza, czy istnieje jej katalog map. Jeżeli wyłączasz usługę dla jakiejś domeny NIS, pamiętaj o usunięciu katalogu.

Mapy zwykle są przechowywane w plikach DBM, aby przyśpieszyć poszukiwania. Tworzone są na podstawie plików głównych za pomocą programu *makedbm* (dla serwera Tobiasa) lub *dbmload* (dla serwera Petera).

Przekształcanie pliku głównego do postaci, którą może zrozumieć *dbmload*, zwykle wymaga pewnych magicznych poleceń *awk* i *sed*, które są męczące przy wpisywaniu i trudne do zapamiętania. Dlatego pakiet Petera Erikssona, *ypserv*, zawiera plik Makefile (nazwany *ypMakefile*), który realizuje tę konwersję dla większości plików głównych. Powinieneś zainstalować go pod nazwą *Makefile* w katalogu map i dokonać edycji, aby uwzględniał mapy NIS-a, które chcesz współdzielić. Na początku pliku znajdziesz cel `all`, który zawiera usługi oferowane przez *ypserv*. Domyślnie wiersz wygląda tak:

```
all: ethers hosts networks protocols rpc services passwd group netid
```

Jeżeli na przykład nie chcesz generować map *ethers.byname* i *ethers.byaddr*, po prostu usuń `ethers` z tej reguły. Aby przetestować swoją konfigurację, możesz zacząć od jednej lub dwóch map, na przykład *services.**.

Po edycji pliku *Makefile*, będąc w katalogu map, wpisz **`make`**. Spowoduje to automatyczne wygenerowanie i zainstalowanie map. Musisz pamiętać o ich uaktualnianiu po każdej zmianie dokonanej w plikach głównych. W przeciwnym razie zmiany nie będą widoczne w sieci.

Podrozdział *Konfigurowanie klienta NIS z GNU libc* wyjaśnia, jak skonfigurować kod klienta NIS. Jeżeli twoja konfiguracja nie działa, powinieneś spróbować dowiedzieć się, czy twoje zapytania docierają do serwera. Jeżeli podasz opcję *--debug* w wywołaniu polecenia *ypserv*, wypisze ono na konsoli komunikaty o wszystkich przychodzących zapytaniach NIS i zwracanych wynikach. Tam powinieneś znaleźć wskazówkę, gdzie leży problem. Serwer Tobiasa nie ma tej opcji.

Bezpieczeństwo serwera NIS

Z punktu widzenia bezpieczeństwa NIS ma podstawową wadę: plik *passwd* jest dostępny praktycznie dla każdego w całym Internecie, czyli również dla sporej liczby potencjalnych intruzów. Kiedy intruz wie, jak nazywa się twoja domena NIS, i zna adres serwera, może po prostu wysłać zapytanie o mapę *passwd.byname* i natychmiast uzyska wszystkie hasła z twojego systemu (w postaci zaszyfrowanej). Z pomocą szybkiego programu do łamania haseł, jak *crack*, i dobrego słownika, odgadnięcie haseł, przynajmniej kilku użytkowników, nie stanowi problemu.

Tu właśnie przydaje się opcja *securenets*. Na podstawie adresów IP lub numerów sieci zezwala na dostęp do twojego serwera NIS tylko pewnym hostom. Ostatnia wersja *ypserv* implementuje tę funkcję na dwa sposoby. Pierwszy opiera się na specjalnym pliku konfiguracyjnym */etc/ypserv.securenets*, a drugi używa odpowiednich plików */etc/hosts.allow* i */etc/hosts.deny*, które omówiliśmy w rozdziale 12, *Ważne funkcje sieciowe**. Aby więc ograniczyć dostęp do hostów z browaru, administrator sieci powinien dodać poniższy wiersz do pliku *hosts.allow*:

```
ypserv: 172.16.2.
```

Pozwoli to wszystkim hostom o adresach IP **172.16.2.0** na dostęp do serwera NIS. Aby zabronić dostępu wszystkim pozostałym hostom, musisz umieścić w pliku *hosts.deny* następujący wpis:

```
ypserv: ALL
```

Numery IP nie są jedynym sposobem na określenie hostów (lub sieci) w pliku *hosts.allow* i *hosts.deny*. Szczegóły znajdziesz na stronie podręcznika *hosts_access(5)* w twoim systemie. Jednak pamiętaj, że *nie możesz* używać nazw hosta lub domen we wpisie `ypserv`. Jeżeli podasz nazwę hosta, serwer będzie próbował ją rozwiązać, ale resolver z kolei wywoła *ypserv* i wejdziesz w niekończoną pętlę.

Aby skonfigurować bezpieczeństwo `securenets` za pomocą metody */etc/ypserv.securenets*, musisz stworzyć plik konfiguracyjny */etc/ypserv.securenets*. Ten plik konfiguracyjny ma prostą strukturę. Każdy wiersz opisuje host lub sieć, które mają dostęp do serwera. Wszelkie adresy nie opisane przez wpisy w tym pliku nie będą miały dostępu do serwera. Wiersz rozpoczynający się do znaku # będzie traktowany jako komentarz. Przykład 13-1 pokazuje, jak wygląda prosty plik */etc/ypserv.securenets*.

---

* Włączenie metody */etc/hosts.allow* może wymagać ponownej kompilacji serwera. Przeczytaj instrukcje zawarte w pliku *README* dołączonym do pakietu dystrybucyjnego.

Przykład 13-1. Przykładowy plik ypserv.securenets

```
# dopuszczenie po□□cze□ z lokalnego hosta - potrzebne
host 127.0.0.1
# to samo co 255.255.255.255 127.0.0.1
#
# dopuszczenie po□□cze□ z dowolnego hosta z sieci browaru
# wirtualnego
255.255.255.0 172.16.1.0
#
```

Pierwszy wpis w każdym wierszu to maska sieci, a słowo kluczowe `host` jest traktowane jako "maska sieci 255.255.255.255”. Drugi wpis w każdym wierszu to adres IP, którego dotyczy maska sieci.

Trzecia możliwość to użycie bezpiecznego portmappera zamiast *securenets* w *ypserv*. Bezpieczny portmapper (*portmap-5.0*) wykorzystuje także schemat *hosts.allow*, ale udostępnia go dla wszystkich serwerów RPC, a nie tylko dla *ypserv**. Jednak nie powinieneś używać jednocześnie opcji *securenets* i bezpiecznego portmappera, ponieważ spowoduje to nadmiar uwierzytelniania.

Konfigurowanie klienta NIS z GNU libc

Opiszemy teraz i omówimy konfigurację klienta NIS-a wykorzystującego bibliotekę GNU *libc*.

Pierwszym krokiem powinno być powiadomienie klienta NIS, którego serwera ma użyć. Wspomnieliśmy wcześniej, że to *ypbind* dla Linuksa pozwala ci na skonfigurowanie właściwego serwera NIS. Domyślne zachowanie polega na szukaniu serwera w sieci lokalnej. Jeżeli istnieje prawdopodobieństwo, że konfigurowany host (na przykład laptop) będzie przenoszony z jednej domeny do drugiej, powinieneś pozostawić plik */etc/yp.conf* pusty i poszukiwać serwera w sieci lokalnej.

Bezpieczniejsza konfiguracja hostów polega na ustawieniu nazwy serwera w pliku konfiguracyjnym */etc/yp.conf*. Bardzo prosty plik dla hosta z sieci winiarni może wyglądać tak:

```
# yp.conf - Konfiguracja YP dla biblioteki GNU libc
#
ypserver vbardolino
```

Dyrektywa `ypserver` mówi twojemu hostowi, by używał podanej nazwy jako serwera NIS dla domeny lokalnej. W tym przykładzie podaliśmy serwer NIS **vbardolino**. Oczywiście w pliku *hosts* musi znajdować się adres IP odpowiadający **vbardolino**. Możesz również użyć samego adresu IP jako argumentu `server`.

W postaci pokazanej w przykładzie polecenie *ypserver* informuje *ypbind*, aby używało serwera o zadanej nazwie bez względu na to, jaka jest aktualna domena NIS. Jeżeli jednak często przenosisz swój komputer pomiędzy różnymi domenami NIS, warto zachować w pliku *yp.conf* informacje o serwerach dla kilku domen. Umiesz-

* Bezpieczny portmapper jest dostępny przez anonimowy serwer FTP pod adresem **ftp.win.tue.nl** w katalogu */pub/security/*

czasz je tam za pomocą dyrektywy `domain`. Na przykład mógłbyś zmienić poprzedni przykładowy plik, aby dla laptopa wyglądał tak:

```
# yp.conf - Konfiguracja YP dla biblioteki GNU libc
#
domain winery server vbardolino
domain brewery server vstout
```

Pozwala ci to podłączyć laptopa do dowolnej z dwóch domen przez ustawienie w czasie inicjacji komputera odpowiedniej domeny NIS poleceniem *domainname*. Klient NIS używa serwera odpowiedniego dla danej domeny.

Istnieje trzecia możliwość, która może się przydać. Dotyczy sytuacji, gdy nie znasz ani nazwy, ani adresu IP serwera używanego w określonej domenie, ale obstajesz przy używaniu stałej nazwy w pewnych domenach. Wyobraźmy sobie, że chcemy wymusić korzystanie z zadanego serwera w domenie winiarni, ale w domenie browaru chcemy szukać serwera w sieci. Moglibyśmy zmodyfikować nasz plik *yp.conf* do takiej postaci:

```
# yp.conf - Konfiguracja YP dla biblioteki GNU libc
#
domain winery server vbardolino
domain brewery broadcast
```

Słowo kluczowe `broadcast` mówi *ypbind*, by używał w domenie dowolnego, znalezionego serwera NIS.

Po stworzeniu tego podstawowego pliku konfiguracyjnego i upewnieniu się, że jest on czytelny dla wszystkich, powinieneś wykonać swój pierwszy test podłączenia się do serwera. Sprawdź, czy korzystasz z map rozpowszechnianych przez twój serwer, jak *hosts.byname*, i spróbuj je odczytać za pomocą narzędzia *ypcat*:

```
# ypcat hosts.byname
172.16.2.2      vbeaujolais.vbrew.com   vbeaujolais
172.16.2.3      vbardolino.vbrew.com    vbardolino
172.16.1.1      vlager.vbrew.com        vlager
172.16.2.1      vlager.vbrew.com        vlager
172.16.1.2      vstout.vbrew.com        vstout
172.16.1.3      vale.vbrew.com          vale
172.16.2.4      vchianti.vbrew.com      vchianti
```

Uzyskany wynik powinien przypominać to, co pokazaliśmy powyżej. Jeżeli pojawił się komunikat błędu o treści `Can't bind to server which serves domain`, to albo ustawiona przez ciebie nazwa domeny NIS nie ma serwera zdefiniowanego w pliku *yp.conf*, albo serwer z jakiegoś powodu jest nieosiągalny. W tym drugim przypadku sprawdź, czy *ping* do hosta daje pozytywny wynik, i czy serwer NIS rzeczywiście działa. Możesz to sprawdzić, używając polecenia *rpcinfo*, które powinno pokazać następujący wynik:

```
# rpcinfo -u serwer ypserv
program 100004 version 1 ready and waiting
program 100004 version 2 ready and waiting
```

Wybór odpowiednich map

Jeżeli jesteś w stanie skomunikować się z serwerem NIS, musisz zdecydować, które pliki konfiguracyjne zastąpić innymi, a do których dodać mapy NIS-a. Przeważnie będziesz chciał używać dwóch map NIS-a: do przeszukiwania hostów i do przeszukiwania haseł. Pierwsza jest szczególnie przydatna, jeżeli nie masz usługi nazewniczej BIND. Druga pozwala na logowanie się wszystkich użytkowników na konta z dowolnego systemu należące go do domeny NIS. Ma to szczególne znaczenie, jeżeli posiadasz centralny katalog */home*, który hosty współdzielą przez NFS. Mapa haseł zostanie szczegółowo omówiona w następnym podrozdziale.

Inne mapy, takie jak *services.byname*, nie dają tak znacznych korzyści, ale pozwalają zaoszczędzić nieco pracy edycyjnej. Mapa *services.byname* jest cenna, jeżeli instalujesz jakieś aplikacje sieciowe wykorzystujące usługi o nazwach, których nie ma w standardowym pliku *services*.

Zapewne chciałbyś mieć jakiś wybór pomiędzy tym, czy funkcja wyszukiwania używa plików lokalnych, zapytuje serwer NIS, czy używa innych serwerów, jak np. DNS. GNU libc pozwala ci skonfigurować kolejność, w której funkcja uzysku je dostęp do tych usług. Jest to kontrolowane przez plik */etc/nsswitch.conf*, którego nazwa jest skrótem od *Name Service Switch*. Plik ten oczywiście nie jest ograniczony do usługi nazewniczej. Może zawierać wiersze dla dowolnych usług wyszukiwania danych, obsługiwanych przez GNU *libc*.

Poprawna kolejność usług zależy od typu danych, oferowanych przez każdą z usług. Jest mało prawdopodobne, by mapa *services.byname* zawierała wpisy różniące się od tych w lokalnym pliku *services*, będzie jedynie miała dodatkowe wpisy. A więc sensowne wydaje się korzystanie z pliku lokalnego w pierwszej kolejności, a sprawdzanie NIS-a tylko wtedy, gdy nazwa usługi nie zostanie znaleziona. Z drugiej strony informacje o nazwie hosta mogą się często zmieniać, a więc serwer DNS lub NIS powinien zawsze mieć aktualne informacje, natomiast plik *hosts* służy jedynie jako kopia zapasowa na wypadek, gdyby DNS lub NIS nie działały. Dlatego w przypadku nazw hostów, zwykle plik lokalny sprawdza się na końcu.

Poniższy przykład pokazuje, jak wymusić na funkcjach *gethostbyname* i *gethostbyaddr*, by najpierw korzystały z NIS i DNS, a potem dopiero z pliku *hosts*, oraz jak sprawić, by funkcja *getservbyname* najpierw korzystała z plików lokalnych, a dopiero potem z NIS-a. Te funkcje resolvera będą kolejno wypróbowywały każdą z podanych usług. Jeżeli wyszukiwanie się powiedzie, zostanie zwrócony wynik. W przeciwnym razie zostanie sprawdzona kolejna usługa z listy. Konfiguracja pliku dla takiej kolejności wygląda następująco:

```
# ma□y przyk□adowy plik /etc/nsswitch.conf
#
hosts:   nis dns files
services:   files nis
```

Poniżej pokazano pełną listę usług i lokalizacji, które mogą być używane we wpisie w pliku *nsswitch.conf*. Rzeczywiste mapy, pliki, serwery i obiekty są zapytywane

w zależności od nazwy wpisu. Elementy z poniższej listy mogą pojawiać się za dwukropkiem:

`nis`
: Zastosowanie serwera aktualnej domeny NIS. Lokalizacja serwera jest definiowana w pliku *yp.conf*, zgodnie z opisem w poprzednim podrozdziale. W przypadku wpisu `hosts` przeszukiwane są mapy *hosts.byname* i *hosts.byaddr*.

`nisplus` *lub* `nis+`
: Zastosowanie serwera NIS+ dla tej domeny. Lokalizacja serwera jest ustalana na podstawie pliku */etc/nis.conf*.

`dns`
: Zastosowanie serwera nazw DNS. Ten typ usługi jest przydatny tylko we wpisie `hosts`. Wykorzystywane serwery nazwy są wciąż określane na podstawie standardowego pliku *resolv.conf*.

`files`
: Zastosowanie pliku lokalnego, na przykład */etc/hosts*, dla wpisu `hosts`.

`compat`
: Kompatybilność ze starymi formatami plików. Ta opcja może być przydatna, gdy do poszukiwań NIS lub NIS+ jest używany NYS lub glibc 2.x. Choć normalnie te wersje nie potrafią interpretować starszych wpisów NIS w plikach *passwd* i *group*, opcja `compat` pozwala działać im z tymi formatami.

`db`
: Poszukiwanie informacji w plikach DBM umieszczonych w katalogu */var/db*. Nazwa pliku jest taka sama jak odpowiadająca jej mapa NIS.

Aktualnie obsługa NIS-a w GNU *libc* dotyczy następujących baz danych *nsswitch.conf*: `aliases`, `ethers.group`, `hosts`, `netgroup`, `network`, `passwd`, `protocols`, `publickey`, `rpc`, `services` i `shadow`. Prawdopodobnie zostaną dodane następne wpisy.

Przykład 13-2 pokazuje bardziej złożone wykorzystanie innej funkcji pliku *nsswitch.conf*. Słowo kluczowe `[NOTFOUND=return]` we wpisie `hosts` informuje klienta NIS, by kończył poszukiwanie, jeżeli żądany element nie zostanie znaleziony w bazach NIS lub DNS. To znaczy, że klient NIS będzie kontynuował przeszukiwanie plików lokalnych *tylko* wtedy, gdy przeszukiwanie serwerów NIS i DNS się nie uda z jakiegoś innego powodu. Pliki lokalne będą używane jedynie w czasie inicjacji i jako kopia zapasowa w sytuacji, gdy serwer NIS nie działa.

Przykład 13-2. Przykładowy plik nsswitch.conf

```
# /etc/nsswitch.conf
#
hosts:      nis dns [NOTFOUND=return] files
networks:   nis [NOTFOUND=return] files
services:   files nis
protocols:  files nis
rpc:        files nis
```

Biblioteka GNU *libc* pozwala na inne możliwe działania. Opisano to na stronach podręcznika elektronicznego *nsswitch*.

Korzystanie z map passwd i group

Jednym z głównych zastosowań NIS-a jest synchronizowanie informacji o kontach i użytkownikach na wszystkich hostach w domenie NIS. W rezultacie zwykle posiadasz tylko lokalny plik */etc/passwd*, do którego są dodawane informacje z map NIS. Jednak proste włączenie przeszukiwania NIS dla tej usługi w pliku *nsswitch.conf* nie wystarczy.

Jeżeli chcesz się opierać na informacjach o hasłach rozpowszechnianych przez NIS-a, najpierw musisz sprawdzić, czy ID użytkowników w twoim lokalnym pliku *passwd* są zgodne z ID użytkowników widzianych przez serwer NIS-a. Spójność ID użytkowników jest także istotna dla innych celów, jak montowanie wolumenów NFS z innych hostów w twojej sieci.

Jeżeli jakiś numer ID w plikach */etc/passwd* lub */etc/group* różni się od ID zawartego w mapach, musisz poprawić prawa własności wszystkich plików danego użytkownika. Najpierw powinieneś zmienić wszystkie wartości `uid` i `gid` w plikach *passwd* i *group* na nowe, a następnie sprawdzić, czy wszystkie pliki należące do użytkownika mają poprawne prawa i ewentualnie zmienić ich właściciela. Załóżmy, że **news** ma ID użytkownika 9, a **okir** miał przed zmianą ID użytkownika 103. Jako root mógłbyś wydać następujące polecenia:

```
# find / -uid 9 -print >/tmp/uid.9
# find / -uid 103 -print >/tmp/uid.103
# cat /tmp/uid.9 | xargs chown news
# cat /tmp/uid.103 | xargs chown okir
```

Ważne, byś wydał te polecenia po zainstalowaniu nowego pliku *passwd* i byś poznał wszystkie nazwy plików, zanim zmienisz prawa własności jakiegokolwiek z nich. Aby uaktualnić prawa własności plików dla grupy, użyj podobnej metody, ale zamiast `uid`, zastosuj `gid`, a zamiast `chown` – `chgrp`.

Gdy to zrobisz, numery `uid` oraz `gid` w twoim systemie będą zgodne z numerami na pozostałych hostach w twojej domenie NIS. Kolejnym krokiem jest dodanie wierszy konfiguracyjnych do *nsswitch.conf*, pozwalających na wyszukiwanie informacji o użytkownikach i grupach w NIS-ie.

```
# /etc/nsswitch.conf - obs□uga passwd i group
passwd:   nis files
group:    nis files
```

Od tego zależy, gdzie polecenie *login* i wszystkie pochodne szukają informacji o użytkowniku. Gdy użytkownik próbuje się zalogować, *login* pyta najpierw mapy NIS, a jeżeli to poszukiwanie się nie uda, powraca do plików lokalnych. Zwykle z plików lokalnych usuwasz prawie wszystkich użytkowników i pozostawiasz jedynie wpisy dla **root** i kont ogólnych, takich jak **mail**. Robi się tak dlatego, że istotne zadania systemowe wymagają przetłumaczenia wartości `uid` na nazwy użytkownika lub odwrotnie. Na przykład zadania administracyjne *cron* mogą wykonywać polece-

nie *su*, by tymczasowo uzyskać prawa użytkownika **news**, lub pod system UUCP może wysyłać raport o stanie. Jeżeli **news** i **uucp** nie będą miały wpisów w lokalnym pliku *passwd*, zadania nie przejdą nawet przez etap przeszukiwań NIS-a.

Jeżeli używasz też starej implementacji NIS-a (obsługiwanej przez tryb `compat` dla plików *passwd* i *group* w implementacjach NYS lub glibc), musisz wstawić w plikach *passwd* dziwne wpisy specjalne. Wpisy te informują, gdzie zostaną wstawione rekordy NIS w bazie informacji. Wpisy mogą zostać dodane w dowolnym miejscu pliku, zwykle na jego końcu. Wpisy dodawane do pliku */etc/passwd* są następujące:

```
+::::::
```

a do pliku */etc/groups*:

```
+:::
```

Zarówno w glibc 2.x, jak i w NYS możesz zmieniać parametry w rekordzie użytkownika uzyskanym z serwera NIS, tworząc wpisy ze znakiem + umieszczonym przed nazwą użytkownika, i usuwać pewne wpisy, dodając znak -. Na przykład wpisy:

```
+stuart::::::/bin/jacl
-jedd::::::
```

unieważnią powłokę zdefiniowaną dla użytkownika **stuart** na serwerze NIS i nie pozwolą użytkownikowi **jedd** zalogować się na tej maszynie. Pola puste, nie wypełnione, oznaczają wykorzystanie informacji dostarczonych przez serwer NIS.

Są tu jednak dwa ważne zastrzeżenia. Po pierwsze, opisana do tej pory konfiguracja działa tylko dla logowania, które nie wykorzystuje haseł shadow. Zawiłości korzystania z haseł shadow w NIS-ie zostaną omówione w kolejnym podrozdziale. Po drugie, polecenia logowania nie są jedynymi, które dostają się do pliku *passwd* – porównaj polecenie *ls*, które go wciąż używa wiele osób. W pełnym listingu *ls* wyświetla nazwy symboliczne użytkownika i grupy, którzy są właścicielami pliku. To znaczy, że w przypadku napotkania każdego uid i gid polecenie musi zadać zapytanie do serwera NIS. Zapytanie NIS trwa nieco dłużej, niż równoważne przeszukiwanie pliku lokalnego. Może się okazać, że umieszczenie w NIS-ie informacji z plików *passwd* i *group* znacznie zmniejsza wydajność programów, które często korzystają z tych informacji.

To jeszcze nie wszystko. Wyobraź sobie, co się stanie, jeżeli użytkownik zechce zmienić hasło. Zwykle wywołuje on polecenie *passwd*, które wczytuje nowe hasło i uaktualnia lokalny plik *passwd*. Jest to niemożliwe w przypadku NIS-a, gdyż plik nie jest nigdzie dostępny lokalnie, a logowanie się użytkowników do serwera NIS za każdym razem, gdy chcą oni zmienić hasło, nie jest także dobrym rozwiązaniem. Dlatego NIS udostępnia zastępczę *passwd* – polecenie *yppasswd*, które obsługuje zmianę hasła w NIS-ie. Aby zmienić hasło na serwerze, łączy się ono przez RPC z demonem *yppasswdd* na tym serwerze i przekazuje mu aktualne informacje o haśle. Zwykle instaluje się *yppasswd*, zamiast normalnego polecenia *passwd*, w następujący sposób:

```
# cd /bin
# mv passwd passwd.old
# ln yppasswd passwd
```

W tym samym czasie musisz zainstalować na serwerze program *rpc.yppasswdd* i uruchomić go ze skryptu sieciowego. Spowoduje to ukrycie przed użytkownikami większości działań NIS-a.

Używanie NIS-a z obsługą haseł shadow

Korzystanie z NIS-a w połączeniu z plikami haseł shadow jest nieco kłopotliwe. Najpierw złe wiadomości: NIS podważa sens używania haseł shadow. Schemat haseł *shadow* został zaprojektowany po to, by zabronić użytkownikom nieposiadającym prawa roota dostępu do zaszyfrowanej postaci haseł. Używanie NIS-a do współdzielenia danych shadow powoduje konieczność udostępniania zaszyfrowanych haseł osobom, które podsłuchują odpowiedzi serwera NIS w sieci. Rozwiązanie polegające na zmuszaniu ludzi do wyboru „dobrych" haseł jest zdecydowanie lepsze, niż próba używania haseł shadow w środowisku NIS. Przyjrzyjmy się szybko, jak to zrobić.

W libc5 nie ma prawdziwego rozwiązania pozwalającego na współdzielenie danych shadow za pomocą NIS-a. Jedynym sposobem na rozpowszechnianie informacji o użytkownikach i hasłach przez NIS jest użycie standardowych map *passwd.* *. Jeżeli masz zainstalowane hasła typu shadow, najprostszym sposobem na ich rozsyłanie jest generowanie odpowiedniego pliku *passwd* na podstawie */etc/shadow* za pomocą narzędzi takich, jak *pwuncov*, i tworzenie map NIS-a na podstawie uzyskanego pliku.

Oczywiście wymyślono kilka zabiegów, które umożliwiają używanie haseł shadow i NIS-a w tym samym czasie, jak na przykład instalacja pliku */etc/shadow* na każdym hoście w sieci i dystrybuowanie informacji o użytkownikach przez NIS-a. Jednak ta sztuczka jest naprawdę prymitywna i w zasadzie zaprzecza istocie NIS-a, który ma ułatwiać administrowanie systemem.

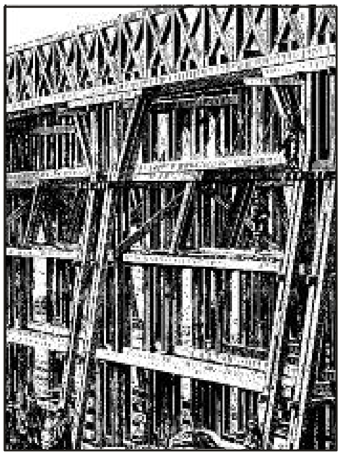
Obsługa NIS-a w bibliotece GNU *libc* (libc6) uwzględnia hasła typu shadow. Nie zapewnia żadnego rozwiązania, które udostępniałoby hasła, ale upraszcza zarządzanie hasłami w środowiskach, w których chcesz używać i NIS-a, i haseł shadow. Aby to zrobić, musisz stworzyć bazę danych *shadow.byname* i dodać poniższy wiersz do swojego pliku */etc/nsswitch.conf*:

```
# Obsługa haseł typu shadow
shadow:   compat
```

Jeśli używasz haseł shadow wraz z NIS-em, musisz przestrzegać pewnej zasady bezpieczeństwa i ograniczyć dostęp do bazy danych NIS. Przypomnij sobie podrozdział *Bezpieczeństwo serwera NIS* z tego rozdziału.

14

Sieciowy system plików



Sieciowy system plików (*Network File System* – NFS) jest prawdopodobnie najbardziej znaną usługą opartą na RPC. Pozwala ona na dostęp do plików znajdujących się na hoście zdalnym dokładnie w taki sam sposób, w jaki masz dostęp do plików lokalnych. Taki dostęp stał się możliwy dzięki połączeniu procedur obsługi w jądrze i demonów przestrzeni użytkownika po stronie klienta z serwerem NFS po stronie serwera. Jest on zupełnie przezroczysty dla klienta i działa pomiędzy różnymi architekturami serwera i hosta.

NFS oferuje szereg przydatnych funkcji:

- Dane wykorzystywane przez wszystkich użytkowników mogą być przechowywane na centralnym hoście, którego katalogi klienty montują w czasie uruchamiania. Na przykład możesz przechowywać wszystkie konta użytkowników na jednym hoście, z którego będziesz montować katalog */home* na wszystkich pozostałych. Jeżeli NFS jest zainstalowany wraz z NIS-em, użytkownicy mogą logować się do dowolnego systemu i wciąż pracować na jednym zestawie plików.
- Dane zajmujące dużo miejsca na dysku mogą być przechowywane na jednym hoście. Na przykład wszystkie pliki i programy związane z LaTeX-em i METAFONT-em mogą być przechowywane i zarządzane w jednym miejscu.
- Dane administracyjne mogą być przechowywane na osobnym hoście. Nie ma potrzeby używania *rcp* do instalacji tego samego głupiego pliku na dwudziestu różnych komputerach.

Konfigurowanie podstawowych operacji NFS po stronie klienta i serwera nie jest trudne. Omawia to ten rozdział.

NFS dla Linuksa to głównie zasługa Ricka Sladkeya*, który napisał kod NFS-a dla jądra i dużą część serwera NFS. Ten ostatni został opracowany na podstawie serwe-

* Z Rickiem możesz skontaktować się pod adresem *jrs@world.std.com*.

ra **unfsd**, którego autorem jest Mark Shand, i na podstawie serwera NFS **hnfs**, napisanego przez Donalda Beckera.

Przyjrzyjmy się, jak działa NFS. Najpierw klient próbuje zamontować katalog z hosta zdalnego w katalogu lokalnym, tak jakby montował fizyczne urządzenie. Jednak składnia używana do określenia zdalnego katalogu jest inna. Na przykład, aby zamontować */home* z hosta **vlager** w katalogu */user* na hoście **vale**, administrator wydaje następujące polecenie na hoście **vale***:

```
# mount -t nfs vlager:/home /users
```

mount podejmie próbę skomunikowania się przez RPC z demonem *rpc.mountd* działającym na hoście **vlager**. Serwer sprawdzi, czy **vale** ma prawo zamontować żądany katalog. Jeżeli tak, zwróci uchwyt pliku. Ten uchwyt będzie używany we wszystkich kolejnych odwołaniach do plików w katalogu */users*.

Gdy ktoś dostaje się do pliku przez NFS, jądro wysyła wywołanie RPC do *rpc.nfsd* (demona NFS) na maszynie serwera. To wywołanie w parametrach zawiera uchwyt pliku, nazwę pliku, do którego się chcemy dostać, i ID użytkownika oraz grupy tego, kto chce się dostać. Są one wykorzystywane przy ustalaniu praw dostępu do zadanego pliku. Aby uniemożliwić nieuprawnionym użytkownikom odczytywanie lub modyfikowanie plików, ID użytkownika i grupy muszą być takie same na obu hostach.

W większości implementacji uniksowych zarówno po stronie klienta, jak i serwera NFS działa w formie demonów jądra uruchamianych z przestrzeni użytkownika w czasie startu systemu. Są to *demony NFS* (*rpc.nfsd*) na hoście serwera i *demony blokowego wejścia/wyjścia* (*biod*) na hoście klienta. Aby poprawić przepustowość, *biod* realizuje asynchroniczne operacje wejścia/wyjścia za pomocą algorytmów odczytu z wyprzedzeniem (ang. *read-ahead*) i zapisywania z opóźnieniem (ang. *write-behind*). Ponadto kilka demonów *rpc.nfsd* zwykle działa jednocześnie.

Aktualna implementacja NFS-a dla Linuksa różni się od klasycznego NFS-a, w którym kod serwera działa całkowicie w przestrzeni użytkownika, a więc uruchomienie kilku kopii jednocześnie jest nieco bardziej skomplikowane. Aktualna implementacja *rpc.nfsd* oferuje eksperymentalną funkcję pozwalającą na ograniczenie obsługi dla wielu serwerów. W jądrach serii 2.2 Olaf Kirch stworzył serwer NFS oparty na jądrze. Jego wydajność jest znacznie lepsza niż istniejących implementacji opartych na przestrzeni użytkownika. Opiszemy go w dalszej części rozdziału.

Przygotowanie NFS-a

Zanim będziesz mógł użyć NFS-a, czy to serwera, czy klienta, musisz sprawdzić, czy twoje jądro jest skompilowane z jego obsługą. Nowsze jądra mają prosty interfejs oparty na systemie plików */proc*; plik */proc/filesystems*, możesz wyświetlić za pomocą *cat*:

```
$ cat /proc/filesystems
   minix
   ext2
```

* W rzeczywistości możesz pominąć argument `-t nfs`, ponieważ z dwukropka *mount* wnioskuje, że chodzi o wolumen NFS.

```
msdos
nodev proc
nodev nfs
```

Jeżeli na tej liście braku jest `nfs`, musisz skompilować jądro z obsługą NFS-a lub załadować moduł, jeżeli obsługa NFS-a została skompilowana w postaci modułu. Konfigurowanie opcji jądra wyjaśniono w podrozdziale *Konfigurowanie jądra* w rozdziale 3, *Konfigurowanie sprzętu sieciowego*.

Montowanie wolumenu NFS

Montowanie wolumenów NFS przypomina do złudzenia montowanie normalnych systemów plików. Wywołaj *mount*, używając następującej składni:

```
# mount -t nfs wolumen_nfs katalog_lokalny opcje
```

`wolumen_nfs` jest określany następująco: `zdalny_host:zdalny_katalog`. Ponieważ ten zapis jest unikatowy dla systemów plików NFS, możesz nie stosować opcji *–t nfs*.

Istnieje szereg dodatkowych opcji, które możesz podać w poleceniu *mount* przy montowaniu wolumenu NFS. Mogą być one podane zarówno z przełącznikiem *–o* w wierszu poleceń, jak i w polu opcji wpisu */etc/fstab* dla wolumenu. W obu przypadkach opcje są oddzielone przecinkami i nie mogą zawierać białych znaków. Opcje określone w wierszu poleceń zawsze mają wyższy priorytet, niż te podane w pliku *fstab*.

Oto przykładowy wpis w pliku */etc/fstab*:

```
# wolumen             punkt montowania   typ   opcje
news:/var/spool/news   /var/spool/news    nfs   timeo=14,intr
```

Z kolei wolumen może zostać zamontowany poleceniem:

```
# mount news:/var/spool/news
```

W przypadku braku wpisu w pliku *fstab*, wywołanie *mount* wygląda dużo gorzej. Na przykład załóżmy, że montujesz katalogi macierzyste swoich użytkowników z komputera o nazwie **moonshot**, który wykorzystuje domyślny rozmiar bloku (4 KB) dla operacji odczytu i zapisu. Za pomocą poniższego polecenia możesz zwiększyć rozmiar bloku do 8 KB, by uzyskać lepszą wydajność:

```
# mount moonshot:/home /home -o rsize=8192,wsize=8192
```

Lista wszystkich dopuszczalnych opcji znajduje się na stronie podręcznika elektronicznego *nfs(5)*. Poniżej pokazano skróconą listę opcji, których prawdopodobnie będziesz najczęściej używać:

rsize=n i *wsize=n*

Określają rozmiar datagramu używanego przez klientów NFS, odpowiednio w żądaniach odczytu i zapisu. Domyślna wartość zależy od wersji jądra, ale zwykle wynosi 1024 bajty.

timeo=n

Wskazuje, ile czasu (w dziesiątych częściach sekundy) klient NFS czeka na zakończenie żądania. Domyślna wartość wynosi 7 (0,7 sekundy). To, co się dzieje po upłynięciu tego czasu, zależy od tego, czy używasz opcji *hard* czy *soft*.

hard

Jawnie oznacza wolumen jako zamontowany na stałe. Jest włączona domyślnie. Opcja powoduje, że serwer zgłasza na konsoli komunikat, gdy nie uda się mu dostać do wolumenu po upłynięciu długiego czasu oczekiwania, i wciąż próbuje się do niego dostać.

soft

Ta opcja (w przeciwieństwie do montowania na stałe) powoduje, że gdy upłynie długi czas oczekiwania, do procesu próbującego wykonać operację na pliku jest zgłaszany błąd wejścia/wyjścia.

intr

Pozwala sygnałom przerywać wywołanie NFS. Przydatne do przerywania działania, jeżeli serwer nie odpowiada.

Wszystkie opcje, poza *rsize* i *wsize*, dotyczą zachowania klienta w sytuacji, gdy serwer jest chwilowo niedostępny. Działają razem w następujący sposób: gdy klient wysyła żądanie do serwera NFS, oczekuje, że operacja zostanie zakończona po zadanym okresie czasu (określonym w opcji *timeout*). Jeżeli przez ten czas nie uzyska potwierdzenia, następuje tak zwany *krótki czas oczekiwania* (ang. *minor timeout*): operacja jest powtarzana, ale teraz jej czas oczekiwania jest dwukrotnie dłuższy. Jeśli wartość czasu oczekiwania dojdzie do 60 sekund, następuje *długi czas oczekiwania* (ang. *major timeout*).

Domyślnie długi czas oczekiwania powoduje, że klient wypisuje na konsoli komunikat i rozpoczyna oczekiwanie od nowa, tym razem pierwszy czas oczekiwania jest dwukrotnie dłuższy od poprzedniego. Potencjalnie ten czas będzie się wydłużać w nieskończoność. Wolumeny, w odniesieniu do których operacje są uparcie wykonywane powtórnie, są nazywane *zamontowanymi na stałe* (ang. *hard-mounted*). W przeciwieństwie do nich, wolumeny *zamontowane nietrwale* (ang. *soft-mounted*) generują błąd wejścia/wyjścia dla wywołującego procesu, gdy wystąpi długi czas oczekiwania. Ponieważ bufor pamięci podręcznej jest zapisywany z opóźnieniem, warunek błędu nie jest dostarczany do samego procesu przed wywołaniem następnej funkcji *write*, a więc program może w ogóle nigdy nie uzyskać pewności, że operacja zapisu na wolumen zamontowany nietrwale zakończyła się poprawnie.

To, czy montujesz wolumeny jako zamontowane na stałe, czy jako nietrwałe zależy w dużej mierze od twoich upodobań, ale także od typu informacji, które się na nich znajdują. Na przykład, jeżeli zamontujesz programy X przez NFS, pewnie nie będziesz chciał, by twoja X-sesja została przerwana dlatego, że ktoś zatrzymał ruch w sieci, bo uruchomił właśnie siedem kopii Dooma lub wyciągnął na chwilę wtyczkę Ethernet. W przypadku wolumenu zamontowanego na stałe masz pewność, że komputer będzie czekał, aż zaistnieje możliwość ponownego skontaktowania się z serwerem NFS. Z drugiej strony, mało potrzebne dane, takie jak zamontowane

przez NFS partycje z grupami dyskusyjnymi czy archiwami FTP, można montować w sposób nietrwały, tak że gdy zdalny host będzie tymczasowo nieosiągalny lub wyłączony, nie zawiesi two jej sesji. Jeżeli połączenie sieciowe z serwerem jest niestabilne lub przechodzi przez obciążony ruter, możesz zwiększyć wstępny czas oczekiwania za pomocą opcji *timeo* lub zamontować wolumen na stałe. Wolumeny NFS są domyślnie montowane na stałe.

Montowanie na stałe stanowi problem, ponieważ domyślnie operacje na pliku nie są przerywalne. Tak więc, jeżeli proces na przykład próbuje zapisać do zdalnego serwera, a ten jest nieosiągalny, aplikacja użytkownika zawiesza się i użytkownik nie może nic zrobić, by przerwać operację. Jeżeli użyjesz opcji *intr* w połączeniu z montowaniem na stałe, wszelkie sygnały odebrane przez proces przerywają wywołanie NFS, tak że użytkownicy mogą wciąż przerwać zawieszone dostępy do plików i pracować dalej (choć bez zapisania pliku).

Zwykle demon *rpc.mountd* w jakiś sposób pilnuje, które katalogi zostały zamontowane i przez jakie hosty. Informację tę można wyświetlić, używając programu *showmount*, który jest także dołączony do pakietu serwera NFS:

```
# showmount -e moonshot
Export list for localhost:
/home <anon clnt>

# showmount -d moonshot
Directories on localhost:
/home

# showmount -a moonshot
All mount points on localhost:
localhost:/home
```

Demony NFS

Jeżeli chcesz udostępnić usługę NFS innym hostom, musisz uruchomić na swoim komputerze demony *rpc.nfsd* i *rpc.mountd*. Jako programy oparte na RPC nie są one zarządzane przez *inetd*, ale są uruchamiane w czasie startu systemu i rejestrują się w portmapperze. Dlatego możesz je uruchamiać tylko, jeżeli masz pewność, że działa *rpc.portmap*. Zwykle w jednym ze swoich skryptów uruchamiających sieć powinieneś mieć coś takiego:

```
if [ -x /usr/sbin/rpc.mountd ]; then
   /usr/sbin/rpc.mountd; echo -n " mountd"
fi
if [ -x /usr/sbin/rpc.nfsd ]; then
   /usr/sbin/rpc.nfsd; echo -n " nfsd"
fi
```

Informacje o prawach własności plików udostępnianych klientom przez demona NFS zwykle zawierają numeryczne wartości ID użytkownika i grupy. Jeżeli zarówno klient, jak i serwer kojarzą te same nazwy użytkownika i grupy z ich wartościami numerycznymi ID, mówi się, że mają wspólną przestrzeń uid/gid. Na przykład jest

tak w sytuacji, gdy używasz NIS-a do rozpowszechniania informacji *passwd* do wszystkich hostów swojej sieci lokalnej.

Jednak w pewnych sytuacjach ID na różnych hostach nie zgadzają się ze sobą. Zamiast uaktualniać uid i gid klienta, tak by pasowały do używanych przez serwer, możesz użyć demona mapowania *rpc.ugidd*, który wyrównarozbieżności w odwzorowaniach. Korzystając z opcji *map_daemon* (wyjaśnionej nieco dalej), możesz zmusić *rpc.nfsd*, by odwzorował przestrzeń uid/gid serwera na przestrzeń uid/gid klienta za pomocą *rpc.ugidd* po stronie klienta. Niestety demon *rpc.ugidd* nie zawsze znajduje się w dystrybucji Linuksa, a więc jeżeli go potrzebujesz, a nie ma go w twojej dystrybucji, będziesz musiał skompilować kody źródłowe.

rpc.ugidd jest serwerem opartym na RPC, uruchamianym ze startowych skryptów sieciowych, tak jak *rpc.nfsd* i *rpc.mountd*.

```
if [ -x /usr/sbin/rpc.ugidd ]; then
   /usr/sbin/rpc.ugidd; echo -n " ugidd"
fi
```

Plik exports

Teraz zobaczymy, jak konfiguruje się serwer NFS. Zwłaszcza przyjrzymy się, jak należy poinformować serwer NFS, które systemy plików powinien udostępniać do montowania. Poznamy też różne parametry, które kontrolują dostęp klientów do systemu plików. Serwer określa typ dostępu do jego plików. W pliku */etc/exports* znajduje się lista systemów plików, które serwer udostępnia klientom do montowania i użytku.

Domyślnie *rpc.mountd* nie pozwala na montowanie wszystkich katalogów, co jest raczej rozsądnym podejściem. Jeśli chcesz pozwolić jednemu lub kilku hostom na montowanie katalogu przez NFS, musisz *wyeksportować host*, to znaczy wpisać go do pliku *exports*. Przykład owy plik może wyglądać tak:

```
# plik exports dla hosta vlager
/home         vale(rw) vstout(rw) vlight(rw)
/usr/X11R6    vale(ro) vstout(ro) vlight(ro)
/usr/TeX      vale(ro) vstout(ro) vlight(ro)
/             vale(rw,no_root_squash)
/home/ftp      (ro)
```

Każdy wiersz definiuje katalog i hosty, które mogą go montować. Nazwa hosta jest zwykle podawana w postaci pełnej nazwy domenowej, ale może dodatkowo zawierać znaki uniwersalne * i ?, które działają w sposób analogiczny do powłoki Bourne'a. Na przykład do `lab*.foo.com` pasuje zarówno **lab01.foo.com**, jak i **laboratory.foo.com**. Host można także wskazać za pomocą zakresu adresów IP w postaci `adres/maska_sieci`. Jeżeli nazwa hosta nie zostanie podana, tak jak w przypadku katalogu */home/ftp* w poprzednim przykładzie, oznacza to, że pasuje każdy host i każdy ma prawo montować katalog.

Przy sprawdzaniu klienta w pliku *exports*, *rpc.mountd* szuka nazwy hosta klienta za pomocą wywołania *gethostbyaddr*. Jeżeli używany jest DNS, to wywołanie zwraca kanoniczną nazwę hosta klienta, a więc musisz pamiętać, by nie używać aliasów

w pliku *exports*. W środowisku NIS zwracaną nazwą jest pierwsza pasująca nazwa z bazy danych hostów. Nigdy – ani w przypadku użycia DNS-a, ani NIS-a – zwracana nazwa nie jest pierwszą nazwą hosta, pasującą do adresu klienta i znalezioną w pliku *hosts*.

Za nazwą hosta następuje opcjonalna lista znaczników ujętych w nawiasy; poszczególne elementy listy są oddzielone przecinkami. Niektóre wartości tych znaczników to:

secure
: Ten znacznik powoduje, że żądania nie musi pochodzić z zarezerwowanego portu źródłowego, tzn. o wartości mniejszej niż 1024. Znacznik ten jest ustawiony domyślnie.

insecure
: Ten znacznik działa odwrotnie niż znacznik *secure*.

ro
: Ten znacznik powoduje, że wolumen NFS jest montowany jako przeznaczony tylko do odczytu. Znacznik jest ustawiony domyślnie.

rw
: Ta opcja montuje pliki do odczytu i zapisu.

root_squash
: Ta funkcja, związana z bezpieczeństwem, odmawia użytkownikom uprzywilejowanym na zadanych hostach specjalnych praw dostępu przez odwzorowanie żądań z uid 0 po stronie klienta na uid 65534 (tzn. –2) po stronie serwera. Ta wartość uid powinna być związana z użytkownikiem **nobody**.

no_root_squash
: Nie odwzorowuje żądań z uid 0. Ta opcja jest ustawiona domyślnie, a więc użytkownicy uprzywilejowani mają nieograniczony dostęp do wyeksportowanych katalogów systemu.

link_relative
: Ta opcja zamienia bezwzględne dowiązania symboliczne (gdzie dowiązania rozpoczynają się od ukośnika) na dowiązania względne. Ta opcja ma sens tylko wtedy, gdy zamontowany jest cały system plików hosta. W przeciwnym razie niektóre dowiązania mogą wskazywać donikąd lub co gorsza, na pliki, których nigdy nie miały wskazywać. Ta opcja jest domyślnie włączona.

link_absolute
: Ta opcja pozostawia dowiązania symboliczne bez zmian (normalne zachowanie serwerów NFS firmy Sun).

map_identity
: Ta opcja mówi serwerowi, by zakładał, że klient używa tych samych wartości uid i gid co serwer. Ta opcja jest ustawiona domyślnie.

map_daemon
: Ta opcja mówi serwerowi NFS, by zakładał, że klient i serwer nie współdzielą tej samej przestrzeni uid/gid. Dlatego *rpc.nfsd* tworzy listę, która odwzorowuje ID

pomiędzy klientem a serwerem, zadając zapytania demonowi *rpc.ugidd* na maszynie klienta.

map_static
: Ta opcja pozwala na podanie nazwy pliku, który zawiera statyczne odwzorowanie wartości uid i gid. Na przykład `map_static=/etc/nfs/vlight.map` wskazywałby plik */etc/nfs/vlight.map* jako plik zawierający odwzorowania uid/gid. Składnia pliku odwzorowań jest opisana na stronie podręcznika elektronicznego *exports(5)*.

maps_nis
: Ta opcja powoduje, że serwer NIS-a realizuje odwzorowania uid i gid.

anonuid i *anongid*
: Te opcje pozwalają na określenie uid i gid kont anonimowych. Jest to przydatne, jeżeli masz publicznie wyeksportowany wolumen.

Wszelkie błędy przy analizie składniowej pliku *exports* są zgłaszane do funkcji `daemon` *sysloga* na poziomie `notice`, o ile są uruchomione demony *rpc.nfsd* i *rpc.mountd*.

Zauważ, że nazwy hostów są uzyskiwane na podstawie adresów IP klienta przez odwzorowanie odwrotne, a więc resolver musi być odpowiednio skonfigurowany. Jeżeli używasz BIND i jesteś świadomy problemów związanych z bezpieczeństwem, powinieneś włączyć w swoim pliku *host.conf* sprawdzanie podszywania się. Te tematy omawiamy w rozdziale 6, *Usługi nazewnicze i konfigurowanie resolvera*.

Serwer NFSv2 oparty na jądrze

Tradycyjnie używany w Linuksie serwer NFS działający w przestrzeni użytkownika jest niezawodny, ale sprawia problemy wydajnościowe, jeżeli jest przeciążony. Dzieje się tak głównie ze względu na obciążenie wnoszone przez interfejs wywołań systemowych, ale też dlatego, że musi on dzielić czas z innymi, potencjalnie mniej istotnymi procesami działającymi w przestrzeni użytkownika.

Jądro 2.2.0 obsługuje ekperymentalny serwer NFS oparty na jądrze, stworzony przez Olafa Kircha i dalej rozwijany przez H.J. Lu, G. Allana Morrisa i Tronda Myklebusta. Serwer NFS oparty na jądrze daje zdecydowaną poprawę wydajności.

W obecnych dystrybucjach możesz znaleźć narzędzia serwera w postaci pakietów. Jeżeli ich tam nie ma, możesz je znaleźć pod adresem *http://csua.berkeley.edu/~gam3/knfsd/*. Aby używać tych narzędzi, musisz skompilować jądro 2.2.0 z demonem NFS opartym na jądrze. Możesz upewnić się, czy twoje jądro ma wbudowanego demona NFS, sprawdzając, czy istnieje plik */proc/sys/sunrpc/nfsd_debug*. Jeżeli go nie ma, możesz załadować moduł *rpc.nfsd*, używając narzędzia *modprobe*.

Demon NFS oparty na jądrze wykorzystuje standardowy plik konfiguracyjny */etc/exports*. Pakiet zawiera zastępcze wersje demonów *rpc.mountd* i *rpc.nfsd*, które uruchamiasz prawie tak samo, jak ich odpowiedniki działające w przestrzeni użytkownika.

Server NFSv3 oparty na jądrze

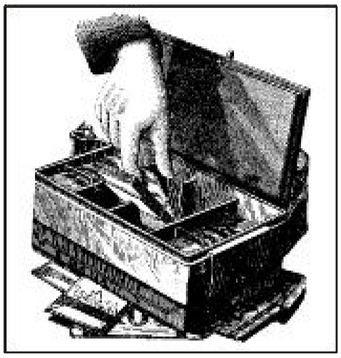
Powszechnie używaną wersją NFS-a jest wersja 2. Jednak technika szybko idzie naprzód i ujawnia niedoskonałości, które może poprawić tylko inna wersja protokołu. Wersja 3 sieciowego systemu plików obsługuje większe pliki i systemy plików, gwarantując znacznie większe bezpieczeństwo i oferując szereg poprawek wydajnościowych, które przydadzą się większości użytkowników.

Olaf Kirch i Trond Myklebust pracują nad eksperymentalnym serwerem NFSv3. Jest on umieszczany w jądrach serii 2.3. Dostępne są łaty dla jądra serii 2.2. Korzysta on z demona NFS w wersji 2, opartego na jądrze.

Łaty są dostępne na stronie macierzystej serwera NFS opartego na jądrze Linuksa, znajdującej się pod adresem *http://csua.berkeley.edu/~gam3/knfsd/*.

15

IPX i system plików NCP



Na długo za nim firma Microsoft zajęła się sieciami i nawet zanim Internet wyszedł poza kręgi akademickie, firmy współdzieliły pliki i drukarki, używając serwerów plików i drukowania opartych na systemie operacyjnym Novell Netware i związanych z nim protokołach*. Wielu z tych użytkowników nadal utrzymuje sieci wykorzystujące te protokoły i chciałoby integrować je z nowszymi sieciami TCP/IP.

Linux obsługuje nie tylko protokoły TCP/IP, ale także zestaw protokołów używanych przez system operacyjny NetWare firmy Novell Corporation. Protokoły te są dalekimi kuzynami TCP/IP i choć pełnią podobną rolę, różnią się pod wieloma względami i niestety nie są ze sobą kompatybilne.

Linux posiada zarówno darmowe, jak i komercyjne oprogramowanie do integracji z produktami Novella.

W tym rozdziale krótko opiszemy same protokoły, a skupimy się na konfigurowaniu i wykorzystaniu darmowego oprogramowania, które pozwala Linuksowi na współpracę z produktami firmy Novell.

Xerox, Novell i historia

Przyjrzyjmy się najpierw, skąd pochodzą protokoły i jak wyglądają. Pod koniec lat siedemdziesiątych w firmie Xerox Corporation opracowano, a następnie opublikowano otwarty standard o nazwie *Xerox Network Specification* (XNS). Standard ten opisywał szereg protokołów obsługujących ogólnie pojętą komunikację sieciową, ze szczególnym uwzględnieniem sieci lokalnych. Były tam wykorzystane dwa główne protokoły sieciowe: internetowy protokół datagramów (*Internet Datagram Protocol* – IDP), zapewniający bezpołączeniowe i zawodne przesyłanie datagramów między hostami, oraz protokół komunikacyjny pakietów danych (*Sequenced Packet Protocol* – SPP) zaadoptowany z IDP, ale połączeniowy i niezawodny. Datagramy w sieci XNS były adresowane indywidualnie. Schemat adresowania opierał się na połączeniu 4-bajtowego adresu sieci IDP (który był unikalnie przypisany do każdego segmentu

* Novell i NetWare są znakami towarowymi firmy Novell Corporation.

sieci lokalnej Ethernet) i 6-bajtowego adresu węzła (adresu karty sieciowej). Rutery były urządzeniami, które przekazywały datagramy pomiędzy dwoma lub kilkoma oddzielnymi sieciami IDP. IDP nie obsługuje podsieci. Każdy nowy zbiór hostów wymaga innego adresu sieci. Adresy sieci są dobierane tak, aby były unikalne w obrębie intersieci. Czasem administratorzy tworzą konwencje, przypisujące typy informacji (np. rejon geograficzny) do poszczególnych bajtów adresu, a więc adresy sieci są przydzielane w systematyczny sposób. Nie jest to jednak wymaganie protokołu.

Firma Novell zdecydowała się oprzeć swoją sieć na pakiecie XNS. Stworzyła kilka rozszerzeń dla IDP i SPP, które otrzymały nazwy: IPX (*Internet Packet eXchange*) i SPX (*Sequenced Packet eXchange*). Dodała też nowe protokoły, takie jak NCP (*NetWare Core Protocol*), pozwalający na współdzielenie plików i drukarek w sieci IPX, czy SAP (*Service Advertisement Protocol*) dający hostom w sieci Novell wiedzę o usługach oferowanych przez poszczególne hosty.

Tabela 15-1 pokazuje powiązanie pomiędzy zestawami protokołów XNS, Novell i TCP/IP pod względem funkcjonalności. Powiązania są jedynie przybliżone, ale powinny pomóc zrozumieć, o co chodzi, gdy będziemy się dalej odwoływać do tych protokołów.

Tabela 15-1. Powiązania między protokołami XNS, Novell i TCP/IP

XNS	*Novell*	*TCP/IP*	*Funkcje*
IDP	IPX	UDP/IP	Bezpołączeniowe i zawodne przesyłanie danych.
SPP	SPX	TCP	Połączeniowe i niezawodne przesyłanie danych.
	NCP	NFS	Usługi plikowe.
	RIP	RIP	Wymiana informacji o rutingu.
	SAP		Wymiana informacji o dostępności usługi.

IPX i Linux

Alan Cox jako pierwszy opracował obsługę protokołu IPX w jądrze Linuksa w 1995 roku*. Na początku przydawało się to w zasadzie tylko do rutowania datagramów IPX. Od tego czasu inni ludzie, głównie Greg Page, rozbudowali tę usługę**. Greg stworzył narzędzia do konfiguracji IPX-a, które wykorzystamy w tym rozdziale do konfigurowania naszych interfejsów. Volker Lendecke opracował obsługę systemu plików NCP, co umożliwiło montowanie w Linuksie wolumenów z podłączonych do sieci serwerów plików NetWare***. Stworzył również narzędzia, które pozwalają drukować do i z Linuksa. Ales Dryak i Martin Stover niezależnie opracowali demony serwera plików NCP dla Linuksa, które pozwalały klientom sieci NetWare montować katalogi linuksowe wyeksportowane jako wolumeny NCP, podobnie jak demon NFS pozwala Linuksowi na udostępnianie klientom systemów plików przez

* Z Alanem można się skontaktować pod adresem *alan@lxorguk.ukuu.org.uk*.

** Z Gregiem możesz się skontaktować pod adresem *gpage@sovereign.org*.

*** Z Volkerem można się skontaktować pod adresem *lendecke@namu01.gwdg.de*.

protokół NFS*. Firma Caldera Systems Inc. oferuje komercyjnego i w pełni licencjonowanego klienta i serwer NetWare obsługujące najnowsze standardy firmy Novell, włącznie z obsługą usług katalogowych NetWare (*NetWare Directory Services* – NDS).

Obecnie Linux obsługuje szeroki zakres usług, które umożliwiają integrowanie systemów z istniejącymi sieciami opartymi na Novellu.

Wsparcie Caldery

Choć w tym rozdziale nie omawiamy szczegółowo wsparcia Caldery dla NetWare, ważniejsze jest to, że w ogóle o tym mówimy. Firma Caldera została założona przez Raya Noorda, który wcześniej był dyrektorem naczelnym firmy Novell. Obsługa NetWare przez Calderę jest produktem komercyjnym i w pełni serwisowanym przez firmę Caldera. Jest to część ich firmowej dystrybucji Linuksa o nazwie Caldera Open Linux. Rozwiązania Caldery są idealnym sposobem na wprowadzenie Linuksa do środowisk, które wymagają zarówno wsparcia komercyjnego, jak i możliwości integracji z istniejącymi lub nowymi sieciami Novell.

Wsparcie Caldery dla NetWare jest w pełni licencjonowane przez firmę Novell, co daje dużą pewność, że produkty obu firm będą ze sobą współpracowały. Dwa wyjątki od tej reguły to: tryb „pure IP" dla klienta oraz serwer NDS (żadna z tych opcji nie była dostępna w czasie pisania tej książki). Dostępne są natomiast zarówno klient, jak i serwer NetWare. Istnieje także zestaw narzędzi, ułatwiających zarządzanie nie tylko serwerami NetWare opartymi na Linuksie, ale także rzeczywistymi serwerami Novell Netware, a to dzięki dużym możliwościom języków skryptowych Uniksa. Więcej informacji na temat Caldery można znaleźć na ich witrynie internetowej**.

Więcej na temat wsparcia NDS-u

W 4. wersji systemu NetWare firma Novell wprowadziła nową funkcję o nazwie usługi katalogowe NetWare (NDS). Specyfikacja NDS-u nie jest dostępna bez specjalnej umowy, co ogranicza rozwój darmowej obsługi tego systemu. Jedyny pakiet *ncpfs* od wersji 2.2.0, który omówimy później, obsługuje NDS. Został on opracowany na zasadzie odwrotnej analizy (ang. *reverse engineering*) protokołu NDS. Wsparcie to wydaje się działać, ale wciąż jest oficjalnie uznawane za eksperymentalne. Z serwerami NetWare 4 możesz używać narzędzi nie-NDS-owych, pracujących w „trybie emulacji bindery".

Oprogramowanie Caldery w pełni obsługuje NDS, ponieważ ich implementacja ma licencję firmy Novell. Ta implementacja nie jest jednak bezpłatna. A więc nie masz dostępu do kodu źródłowego i nie będziesz w stanie dowolnie kopiować i dystrybuować tego oprogramowania.

* Z Alesem można się skontaktować pod adresem *A.Dryak@sh.cvut.cz*, a z Martinem pod adresem *mstover@freeway.de*.

** Informacje na temat firmy Caldera można znaleźć pod adresem *http://www.caldera.com/*.

Konfigurowanie jądra do obsługi IPX-a i NCPFS

Konfigurowanie jądra do obsługi protokołu IPX i systemu plików NCP jest proste. Polega na wybraniu odpowiednich opcji jądra w czasie jego kompilacji. Tak jak dzieje się z wieloma innymi składnikami jądra, elementy IPX i NCPFS mogą być albo wbudowane w jądro, albo skompilowane w postaci modułów i ładowane w razie potrzeby poleceniem *insmod*.

Należy wybrać poniższe opcje, jeżeli chce się mieć w Linuksie obsługę protokołu IPX i rutingu:

```
General setup --->
   [*] Networking support

Networking options --->
   <*> The IPX protocol

Network device support --->
   [*] Ethernet (10 or 100Mbit)
      ... and appropriate Ethernet device drivers

Gdyby□ chcia□, □eby Linux obs□ugiwa□ system plików NCP, tak by mo□na by□o
montowa□ zdalne wolumeny NetWare, musia□by□ dodatkowo wybra□ te opcje:

Filesystems --->
   [*] /proc filesystem support
   <*> NCP filesystem support (to mount NetWare volumes)
```

Gdy skompilujesz i zainstalujesz nowe jądro, jesteś gotów do pracy z IPX-em.

Konfigurowanie interfejsów IPX

Tak jak w TCP/IP, musisz skonfigurować swoje interfejsy IPX, zanim będziesz mógł ich używać. Protokół IPX ma kilka wyjątkowych wymagań. Z tego powodu został stworzony zestaw narzędzi konfiguracyjnych. Będziemy ich używali do konfigurowania naszych interfejsów IPX i rutingu.

Urządzenia sieciowe obsługujące IPX

Protokół IPX zakłada, że hosty, które mogą wymieniać datagramy bez rutowania, należą do tej samej sieci IPX. Wszystkie hosty należące do jednego segmentu Ethernet należą do tej samej sieci IPX. Podobnie (ale mniej intuicyjnie), oba hosty obsługujące łącze szeregowe oparte na PPP muszą należeć do sieci IPX, która sama jest łączem szeregowym. W środowisku Ethernet istnieje szereg różnych typów ramek, które mogą być używane do przenoszenia datagramów IPX. Typy ramek reprezentują różne protokoły Ethernet i opisują różne sposoby przenoszenia wielu protokołów w tej samej sieci Ethernet. Najczęściej będziesz się spotykał z ramkami `802.2` i `ethernet_II`. Więcej o typach ramek powiemy w następnym podrozdziale.

Urządzenia sieciowe Linuksa, które obsługują obecnie protokół IPX, to Ethernet i PPP. Interfejsy Ethernet i PPP muszą być aktywne, zanim nastąpi ich konfiguracja do pracy z protokołem IPX. Zwykle kartę Ethernet konfiguruje się z obsługą zarówno

IP, jak i IPX-a, a więc urządzenie już istnieje. Ale jeżeli chcesz korzystać tylko z sieci IPX, musisz zmienić status urządzenia Ethernet, używając polecenia *ifconfig* w następujący sposób:

```
# ifconfig eth0 up
```

Narzędzia do konfiguracji interfejsu IPX

Greg Page opracował zestaw narzędzi konfiguracyjnych dla interfejsów IPX. Są one rozpowszechniane w postaci pakietów w nowoczesnych dystrybucjach Linuksa, ale można je także uzyskać w postaci źródłowej z anonimowego ośrodka FTP znajdującego się pod adresem *http://metalab.unc.edu* w pliku */pub/Linux/system/filesystems/ncpfs/ipx.tgz*.

Narzędzia IPX są zwykle uruchamiane w czasie startu systemu z pliku *rc*. Twoja dystrybucja może to już robić, jeżeli zainstalowałeś oprogramowanie w postaci pakietów.

Polecenie ipx_configure

Każdy interfejs IPX musi wiedzieć, do której sieci IPX należy i jakiego typu ramki ma używać dla protokołu IPX. Każdy host obsługujący IPX ma przynajmniej jeden interfejs, którego reszta sieci będzie używała do komunikacji z nim. Jest to tak zwany *interfejs podstawowy* (ang. *primary interface*). Obsługa IPX-a w jądrze Linuksa pozwala na automatyczne konfigurowanie tych parametrów. Polecenie *ipx_configure* włącza lub wyłącza tę funkcję automatycznej konfiguracji.

Polecenie *ipx_configure* wywołane bez argumentów wyświetla aktualne ustawienia znaczników automatycznej konfiguracji:

```
# ipx_configure
Auto Primary Select is OFF
Auto Interface Create is OFF
```

Oba znaczniki (`Auto Primary` i `Auto Interface`) domyślnie są wyłączone. Aby je ustawić i włączyć automatyczną konfigurację, po prostu podajesz w wywołaniu następujące argumenty:

```
# ipx_configure --auto_interface=on --auto_primary=on
```

Gdy argument `--auto_primary` ma wartość `on`, jądro automatycznie zapewnia, że przynajmniej jeden aktywny interfejs działa jako interfejs podstawowy dla hosta.

Gdy argument `--auto_interface` ma wartość `on`, sterownik IPX jądra będzie nasłuchiwał wszystkich ramek odebranych na aktywnym interfejsie sieciowym, i będzie próbował ustalić adres sieci IPX oraz używany typ ramki.

Mechanizm automatycznego wykrywania działa bez zarzutu w poprawnie zarządzanych sieciach. Czasem administratorzy sieci idą na skróty i łamią reguły, co może spowodować problemy w kodzie automatycznego wykrywania w Linuksie. Najczęściej dochodzi do nich, gdy jedna sieć IPX jest skonfigurowana do pracy w tej samej sieci Ethernet z wieloma typami ramek. Jest to konfiguracja nieprawidłowa technicznie, gdyż host **802.2** nie może bezpośrednio komunikować się z hostem Et-

hernet-II i dlatego nie mogą one być w tej samej sieci IPX. Oprogramowanie sieciowe IPX Linuksa nasłuchuje na segmencie wysłanych w nim datagramów IPX. Na ich podstawie próbuje zidentyfikować używane adresy sieci i związane z nimi typy ramek. Jeżeli ten sam adres sieci jest używany z wieloma typami ramek lub na wielu interfejsach, kod Linuksa wykrywa kolizję adresów sieci i nie jest w stanie ustalić poprawnego typu ramki. Dowiesz się, że tak się stało, jeżeli w logu systemowym zobaczysz następujące komunikaty:

```
IPX: Network number collision 0x3901ab00
eth0 etherII and eth0 802.3
```

Jeżeli napotkasz taki problem, wyłącz funkcję automatycznego wykrywania i skonfiguruj interfejs ręcznie, używając polecenia *ipx_interface*, które opisujemy poniżej.

Polecenie ipx_interface

Polecenie *ipx_interface* jest używane do ręcznego dodawania, modyfikacji i usuwania protokołu IPX z istniejącego urządzenia sieciowego. Jeżeli nie zadziała właśnie opisana metoda automatycznego wykrywania konfiguracji lub jeżeli nie chcesz pozostawiać konfiguracji interfejsu przypadkowi, powinieneś użyć *ipx_interface*. Pozwala ci ono na podanie adresu sieci IPX, statusu interfejsu podstawowego i typu ramki IPX, która będzie używana przez urządzenie sieciowe. Jeżeli tworzysz wiele interfejsów IPX, dla każdego z nich musisz wykonać polecenie *ipx_interface*.

Składnia tego polecenia przy dodawaniu IPX do istniejącego urządzenia jest prosta i najlepiej wyjaśni ją przykład. Dodajmy IPX do istniejącego urządzenia Ethernet:

```
# ipx_interface add -p eth0 etherII 0x32a10103
```

Oto znaczenie parametrów:

-p
: Ten parametr określa, że ten interfejs powinien być interfejsem podstawowym. Jest on opcjonalny.

eth0
: Jest to nazwa urządzenia sieciowego, do którego dodajemy obsługę IPX.

etherII
: Ten parametr to typ ramki Ethernet II. Może tu wystąpić również wartość: `802.2`, `802.3` lub `SNAP`.

0x32a10103
: To jest adres sieci IPX, do której należy interfejs.

Poniższe polecenie usuwa obsługę IPX z interfejsu:

```
# ipx_interface del eth0 etherII
```

Aby wyświetlić aktualną konfigurację protokołu IPX na urządzeniu sieciowym, użyj:

```
# ipx_interface check eth0 etherII
```

Polecenie *ipx_interface* jest wyjaśnione dokładniej na stronie podręcznika elektronicznego.

Konfigurowanie rutera IPX

Przypomnij sobie z naszego krótkiego omówienia tematu protokołów używanych w środowisku IPX, że IPX jest protokołem rutowalnym i że do rozgłaszania informacji o rutingu jest używany protokół RIP (*Routing Information Protocol*). IPX-owa wersja RIP-a jest podobna do wersji IP. Działają dokładnie w ten sam sposób. Co jakiś czas rutery rozgłaszają zawartość swoich tablic rutingu, a inne rutery „uczą się" jej przez nasłuchiwanie i integrują otrzymane informacje. Hosty muszą tylko znać swoją sieć lokalną i wysyłać datagramy do wszystkich innych sieci przez swój lokalny ruter. Ruter jest odpowiedzialny za przenoszenie tych datagramów i przekazywanie ich do następnego hopa na trasie.

W środowisku IPX w sieci musi być rozgłaszana druga klasa informacji. Protokół ogłaszający usługi (*Service Advertisement Protocol* – SAP) przenosi informacje o rodzajach usług udostępnianych na poszczególnych hostach w sieci. To właśnie protokół SAP pozwala użytkownikom na przykład na uzyskanie listy plików lub serwerów drukowania istniejących w sieci. Protokół SAP działa dzięki hostom, które co jakiś czas rozgłaszają listę udostępnianych usług. Rutery sieciowe IPX zbierają te informacje i propagują je w sieci wraz z informacją o rutingu. Aby ruter mógł zostać uznany za zgodny z protokołem IPX, musi ogłaszać zarówno informacje z RIP-a, jak i z SAP-a.

Tak jak IP, także IPX na Linuksie ma demona rutingu o nazwie *ipxd*, który realizuje zadania związane z zarządzaniem rutingiem. I znów analogicznie do IP, w rzeczywistości to jądro obsługuje przekazywanie datagramów pomiędzy interfejsami sieciowymi IPX, ale w oparciu o zestaw reguł nazywany tablicą rutingu IPX. Demon *ipxd* pilnuje aktualności tego zestawu reguł. Nasłuchuje każdego z aktywnych interfejsów sieciowych i analizuje informacje, żeby wiedzieć, kiedy jest wymagana zmiana w rutingu. Demon *ipxd* odpowiada również na żądania hostów podłączonych bezpośrednio do sieci, które potrzebują informacji o rutingu.

Polecenie *ipxd* jest dostępne w postaci pakietu w niektórych dystrybucjach oraz w postaci źródłowej w anonimowym ośrodku FTP pod adresem *http://metalab.unc.edu/* w pliku */pub/Linux/system/filesystems/ncpfs/ipxripd-x.xx.tgz*.

Demon *ipxd* nie wymaga konfiguracji. Wystarczy go uruchomić, aby automatycznie realizował ruting pomiędzy skonfigurowanymi urządzeniami IPX. Przed uruchomieniem *ipxd* trzeba koniecznie upewnić się, że urządzenia IPX są skonfigurowane poprawnie poleceniem *ipx_interface*. Automatyczne wykrywanie może działać, ale gdy wykonujesz ruting, lepiej nie polegać na przypadku i ręcznie skonfigurować interfejsy, co zaoszczędzi ci bolesnego rozwiązywania trudnych problemów z rutingiem. Co 30 sekund *ipxd* ponownie wykrywa wszystkie lokalnie podłączone sieci IPX i automatycznie nimi zarządza. Pozwala to na zarządzanie sieciami zbudowanymi na interfejsach, które nie utrzymują aktywności przez cały czas, jak interfejsy PPP.

Demon *ipxd* zwykle jest uruchamiany w czasie startu systemu ze skryptu *rc* w następujący sposób:

```
# /usr/sbin/ipxd
```

Nie jest potrzebny znak &, ponieważ *ipxd* sam domyślnie przejdzie do trybu tła. Choć demon *ipxd* najbardziej przydaje się na komputerach działających jako rutery IPX, bywa też użyteczny na hostach podłączonych do segmentów, w których znajduje się wiele ruterów. Jeśli podasz parametr `-p`, *ipxd* będzie działał biernie, nasłuchując informacji o rutingu przychodzących z segmentu i uaktualniając tablice rutingu, ale nie będzie rozsyłał żadnych informacji. W ten sposób host może uaktualniać tablice rutingu i nie żądać za każdym razem informacji o trasie, gdy chce się połączyć z hostem zdalnym.

Statyczny ruting IPX za pomocą polecenia ipx_route

Istnieją sytuacje, kiedy trzeba ustawić ruting IPX „na sztywno". Robimy to podobnie jak dla IP. Polecenie *ipx_route* zapisuje trasę do tablicy rutingu IPX bez potrzeby uczenia się jej od demona *ipxd*. Składnia rutingu jest bardzo prosta (ponieważ IPX nie obsługuje podsieci) i wygląda tak:

```
# ipx_route add 203a41bc 31a10103 00002a02b102
```

Pokazane polecenie dodaje trasę do zdalnej sieci IPX **203a41bc** przez ruter na szej sieci lokalnej **31a10103** o adresie węzła **00002a02b102**.

Adres węzła rutera możesz znaleźć, robiąc prawdziwy użytek z polecenia *tcpdump* z argumentem `-e`, które wyświetla nagłówki poziomu łącza i wskaże ruch z rutera. Jeżeli ruterem jest komputer linuksowy, możesz po prostu użyć polecenia *ifconfig*, by wyświetlić adres.

Za pomocą polecenia *ipx_route* możesz też usunąć trasę:

```
# ipx_route del 203a41bc
```

Trasy aktywne w jądrze możesz wyświetlić, zaglądając do pliku */proc/net/ipx_route*. Nasza dotychczasowa tablica rutingu wygląda tak:

```
# cat ipx_route
Network    Router_Net    Router_Node
203A41BC   31A10103      00002a02b102
31A10103   Directly      Connected
```

Trasa do sieci **31A10103** została stworzona automatycznie przy konfiguracji interfejsu IPX. Każda z naszych sieci lokalnych będzie reprezentowana na przez podobny wpis w pliku */proc/net/ipx_route*. Oczywiście jeżeli nasza maszyna działa jako ruter, musi mieć jeszcze przynajmniej jeden interfejs.

Wewnętrzne sieci IPX i ruting

Hosty IPX o więcej niż jednym interfejsie mają unikalne połączenie adresów sieci/węzła dla każdego ze swoich interfejsów. Aby podłączyć się do takiego hosta, możesz użyć dowolnej kombinacji adresów sieci/węzła. Gdy SAP ogłasza usługi, podaje adres sieci/węzła związany z oferowaną usługą. Na hoście o wielu interfej-

sach oznacza to, że jeden z interfejsów musi być wybrany jako ten, który rozgłasza. Do tego służy znacznik interfejsu podstawowego, o którym mówiliśmy wcześniej. Jest jednak pewien problem: trasa do tego interfejsu nie zawsze może być trasą optymalną, a jeżeli wystąpi awaria sieci, która odizoluje tę sieć od reszty sieci, host stanie się nieosiągalny, nawet jeżeli istnieją inne *możliwe* trasy do innych interfejsów. Inne trasy nigdy nie są znane innym hostom, ponieważ nigdy nie są rozgłaszane i jądro nie ma możliwości dowiedzieć się, że powinno wybrać inny interfejs podstawowy. Aby uniknąć tego problemu, został wymyślony mechanizm, który pozwala, aby host IPX był znany pod jednym, niezależnym od trasy adresem sieci/węzła wykorzystywanym do celów rozgłaszania pakietów SAP. To rozwiązuje nasz problem, gdyż ten nowy adres jest dostępny wszystkim interfejsom hosta i jest jedynym adresem rozgłaszanym przez SAP.

Aby zilustrować problem i jego rozwiązanie, rysunek 15-1 pokazuje serwer podłączony do dwóch sieci IPX, z których jedna ma sieć wewnętrzną. Host na rysunku 15-1 definiuje jeden ze swoich interfejsów jako interfejs podstawowy, załóżmy **0000001a:0800000010aa**. To on zostanie ogłoszony jako punkt dostępu do usługi. Jest to prawidłowe rozwiązanie dla hostów w sieci **0000001a**. Natomiast oznacza, że użytkownicy sieci **0000002c** będą kierowani przez sieć, aby dotrzeć do tego portu, nawet jeżeli port jest bezpośrednio podłączony do sieci, gdyż i tak widzą ten serwer na podstawie tego, co dostali przez protokół SAP.

Jeśli takie hosty będą miały wirtualną sieć o wirtualnych adresach hosta konstruowanych w pełni programowo, problem znika. Sieć wirtualną jest najlepiej wyobrazić sobie jako istniejącą *wewnątrz* hosta IPX. Informacje SAP wystarczy wówczas rozgłaszać jedynie dla adresu sieci/węzła tej sieci wirtualnej. Ta sieć wirtualna jest nazywana *siecią wewnętrzną*. Skąd jednak inne hosty wiedzą, jak dotrzeć do tej sieci wewnętrznej? Hosty zdalne trafiają do sieci wewnętrznej przez sieci, do których host jest podłączony bezpośrednio. Oznacza to, że widzisz wpisy rutingowe odnoszące się do sieci wewnętrznej hostów obsługujących wielokrotne interfejsy IPX.

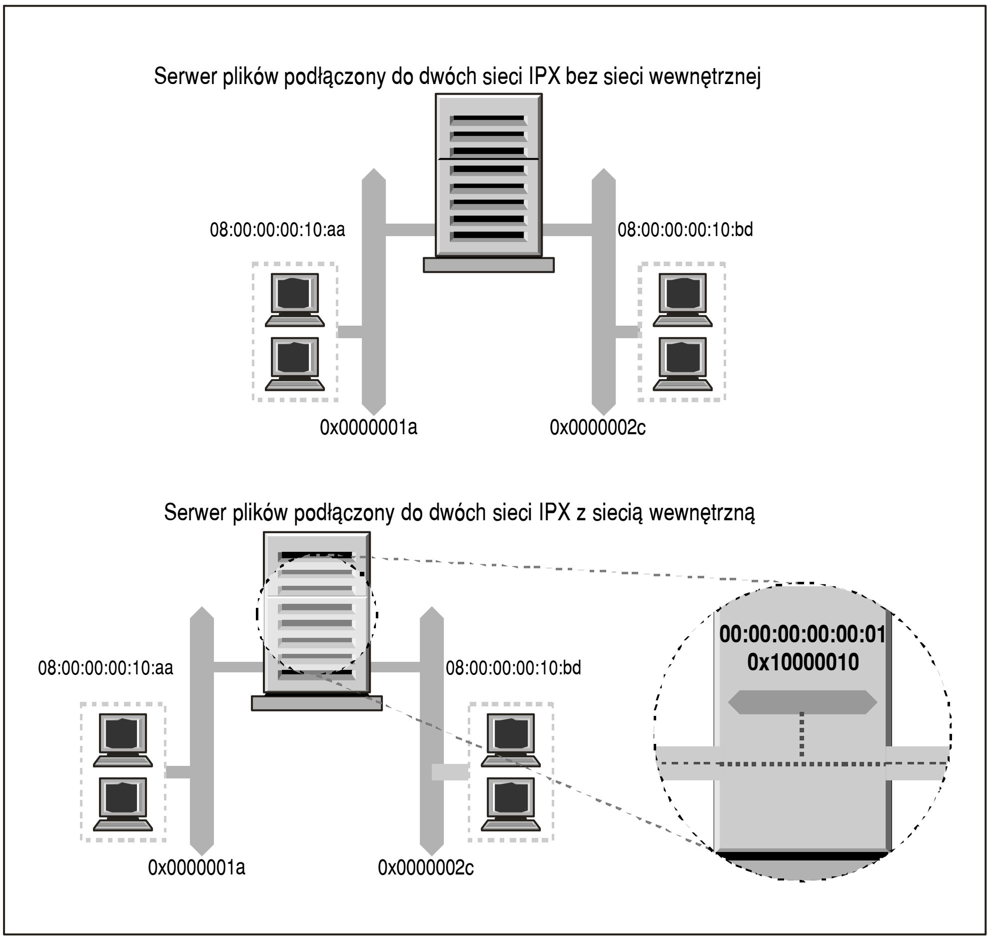
Te trasy powinny być optymalnymi trasami dostępnymi w danej chwili i jeżeli jakaś z nich przestałaby działać, ruting automatycznie zostałby poprowadzony do następnego najlepszego interfejsu i trasy. Na rysunku 15-1 skonfigurowaliśmy wewnętrzną sieć IPX o adresie **0x10000010** i użyliśmy adresu hosta **00:00:00:00:00:01**. Jest to adres naszego interfejsu podstawowego i będzie rozgłaszany przez SAP. Nasz ruting będzie odzwierciedlał tę sieć jako osiągalną przez *któryś* z naszych rzeczywistych portów sieciowych, a więc hosty będą zawsze używały najlepszej trasy do łączenia się z naszym serwerem.

Aby utworzyć taką sieć wewnętrzną, użyj polecenia *ipx_internal_net* znajdującego się w pakiecie narzędzi IPX Grega Page'a. Na przykład tak:

```
# ipx_internal_net add 10000010 000000000001
```

To polecenie tworzy sieć wewnętrzną IPX o adresie **10000010** i adresie węzła **000000000001**. Adres sieci, tak jak każdy inny adres sieci IPX, musi być unikalny w obrębie sieci. Adres węzła jest zupełnie dowolny, gdyż zwykle w sieci jest tylko je-

den wę zeł. Ka żdy host może mieć tyl ko jedną sieć we w nętrzną IPX i je żeli jest ona skon fi gu ro wa na, zaw sze będ zie sie cią pod sta wową.



Ry su nek 15-1. Wewnętrzna sieć IPX

Aby usunąć sieć we wn ętrzną IPX, użyj:

```
# ipx_internal_net del
```

We wn ętrzna sieć IPX ab sol utnie nie jest po trzebna, je żeli tw ój host nie udos tępnia żadn ych usług i ma tyl ko je den ak tywny in terf ejs IPX.

Montowanie zdalnych wolumenów NetWare

IPX jest powszechnie używany do montowania wolumenów NetWare w systemie plików Linuksa. Pozwala to na oparte na plikach współdzielenie danych pomiędzy innymi systemami operacyjnymi i Linuksem. Volker Lendecke opracował klienta NCP dla Linuksa i pakiet narzędzi, które umożliwiają współdzielenie danych.

W środowisku NFS do montowania zdalnego systemu plików użylibyśmy polecenia *mount*. Niestety system plików NCP ma szczególne wymagania, które powodują, że jego wbudowanie w normalne polecenie *mount* jest nierealne. Linux posiada polecenie *ncpmount*, którego użyjemy zamiast *mount*. Polecenie *ncpmount* jest jednym z narzędzi z pakietu *ncpfs* Volkera, dostępnego w większości najnowszych dystrybucji lub w postaci źródłowej pod adresem **ftp.gwdg.de** w katalogu */pub/linux/misc/ncpfs/*. W czasie pisania tej książki obowiązywała wersja 2.2.0.

Zanim będziesz montować wolumeny NetWare, musisz mieć poprawnie skonfigurowany interfejs sieciowy IPX (zgodnie z tym, co opisano wcześniej). Następnie musisz znać szczegóły logowania do serwera NetWare, którego wolumeny chcesz montować. Potrzebne będzie ID użytkownika i hasło. Poza tym musisz wiedzieć, który wolumen chcesz zamontować i w jakim katalogu lokalnym.

Prosty przykład ncpmount

Prosty przykład zastosowania *ncpmount* wygląda tak:

```
# ncpmount -S ALES_F1 -U rick -P d00-b-gud /mnt/brewery
```

To polecenie montuje wszystkie wolumeny z serwera `ALES_F1` w katalogu */mnt/brewery*, wykorzystując użytkownika NetWare `rick` z hasłem `d00-b-gud`.

Polecenie *ncpmount* zwykle ma prawo setuid **root** i dlatego może być używane przez każdego użytkownika Linuksa. Domyślnie użytkownik jest właścicielem połączenia i tylko on lub **root** będzie mógł je odmontować.

NetWare korzysta z pojęcia *wolumenu*, które jest analogiczne do systemu plików w Linuksie. Wolumen NetWare stanowi logiczną reprezentację systemu plików NetWare, który może być pojedynczą partycją dyskową podzieloną na wiele partycji logicznych. Domyślnie NCPFS w Linuksie traktuje wolumeny jako podkatalogi większego logicznego systemu plików reprezentowanego przez cały serwer plików. Polecenie *ncpmount* powoduje, że każdy wolumen NetWare serwera plików jest widoczny jako podkatalog w punkcie montowania. Jest to wygodne, jeżeli chcesz mieć dostęp do całego serwera, ale gdybyś chciał ponownie wyeksportować te katalogi za pomocą NFS-a, niestety nie będziesz w stanie tego zrobić ze skomplikowanych powodów technicznych. Za chwilę omówimy inne, bardziej złożone rozwiązania tego problemu.

Polecenie ncpmount w szczegółach

Polecenie *ncpmount* ma wiele opcji wiersza poleceń, które pozwalają na dużą elastyczność w sposobie montowania NCP. Najważniejsze z nich opisano w tabeli 15-2.

Tabela 15-2. Argumenty polecenia ncpmount

Argument	*Opis*
`-S` *serwer*	Nazwa serwera plików, z którego będą montowane wolumeny.
`-U` *nazwa_u□yt*	ID użytkownika NetWare używane do logowania do serwera plików.
`-P` *has□o*	Hasło używane do zalogowania się do systemu NetWare.
`-n`	Ta opcja musi być użyta w przypadku użytkownika NetWare, który nie posiada hasła.
`-C`	Ten argument wyłącza automatyczną konwersję haseł na pisane dużymi literami.
`-c` *nazwa_klienta*	Ta opcja pozwala ci podać, kto jest właścicielem połączenia z serwerem plików. Jest ona użyteczna na przydrukowaniu w NetWare, które omówimy szczegółowo za chwilę.
`-u` *uid*	ID użytkownika w Linuksie, który powinien być pokazany jako właściciel plików w zamontowanym katalogu. Jeżeli nie zostanie podany, domyślnie będzie to użytkownik, który wywołał polecenie *ncpmount*.
`-g` *gid*	ID grupy w Linuksie, która powinna być pokazana jako właściciel plików w zamontowanym katalogu. Jeżeli nie zostanie podana, domyślnie będzie to ID grupy, do której należy użytkownik wywołujący polecenie *ncpmount*.
`-f` *prawa_dost_plików*	Ta opcja pozwala na ustawienie praw dostępu, jakie powinny mieć pliki w zamontowanym katalogu. Wartości powinny być określane ósemkowo, np. `0664`. Rzeczywiste prawa dostępu są obliczane przez zamaskowanie praw podanych w tej opcji z prawami użytkownika NetWare do plików na serwerze. Musisz mieć prawa na serwerze oraz musisz podać prawa w tej opcji, aby uzyskać dostęp do pliku. Domyślna wartość jest ustalana na podstawie aktualnej wartości `umask`.
`-d` *prawa_dost_katalogów*	Ta opcja pozwala na ustawienie praw dostępu, jakie powinny mieć katalogi w zamontowanym katalogu. Działa w taki sam sposób jak opcja *–f*, z tą różnicą, że domyślne prawa dostępu są ustalane na podstawie aktualnej wartości `umask`. Prawa wykonywania są przydzielane wszędzie tam, gdzie istnieją prawa odczytu.
`-V` *wolumen*	Ta opcja pozwala ci podać nazwę jednego woluminu NetWare, który chcesz zamontować w punkcie montowania. W przeciwnym razie montowane są wszystkie woluminy. Ta opcja jest potrzebna, jeżeli chcesz ponownie wyeksportować przez NFS zamontowane woluminy NetWare.
`-t` *czas_oczekiwania*	Ta opcja pozwala na określenie, ile czasu klient NCPFS czeka na odpowiedź z serwera. Domyślna wartość to 60 milisekund. Czas oczekiwania jest podawany w setnych częściach sekundy. Jeżeli napotkasz na jakieś problemy ze stabilnością woluminów NCP, powinieneś zwiększyć tę wartość.
`-r` *licznik_ponownych_prób*	Kod klienta NCP próbuje wiele razy wysyłać datagramy do serwera, zanim stwierdzi, że połączenie jest nieaktywne. Ta opcja pozwala zmienić domyślną liczbę prób, która wynosi 5.

Ukrywanie twojego hasła użytkownika systemu NetWare

Umieszczanie hasła w wierszu poleceń, jak to robiliśmy w poleceniu *ncpmount*, stwarza pewne zagrożenie. Inni użytkownicy, którzy pracują równocześnie z tobą, mogliby zobaczyć hasło, gdyby uruchomili takie programy, jak *top* czy *ps*. Aby zmniejszyć ryzyko podejrzenia hasła NetWare, *ncpmount* może odczytywać pewne szczegóły z pliku znajdującego się w katalogu macierzystym użytkownika. W tym pliku użytkownik wpisuje swoją nazwę i hasło z nią związane, niezbędne do zalogowania się do serwerów plików, których wolumeny chce montować. Plik nosi nazwę *~/.nwclient* i musi mieć prawa dostępu `0600`, żeby nikt nie mógł go przeczytać. Jeżeli prawa dostępu są niepoprawne, polecenie *ncpmount* nie pozwoli wykorzystać tego pliku.

Plik ma bardzo prostą składnię. Wszelkie wiersze rozpoczynające się od znaku `#` są traktowane jako komentarze i są ignorowane. Pozostałe wiersze mają następującą składnię:

```
serwer_plików/użytkownik hasło
```

`serwer_plików` to nazwa serwera plików zawierającego wolumeny, które chcesz montować. `użytkownik` to nazwa konta użytkownika na serwerze NetWare. Pole `hasło` jest opcjonalne. Jeżeli nie zostanie podane, polecenie *ncpmount* pyta o hasło w czasie montowania wolumenu. Jeżeli w polu `hasło` zostanie podany znak -, konto nie ma hasła. Jest to równoważne argumentowi `-n` wiersza poleceń.

W pliku możesz umieścić dowolną liczbę wierszy, ale pole serwera plików musi być unikalne. Pierwszy wpis w pliku ma szczególne znaczenie. Polecenie *ncpmount* wykorzystuje argument wiersza poleceń `-S`, aby ustalić, którego wpisu z pliku *~/.nwclient* ma używać. Jeżeli serwer nie zostanie określony przez opcję `-S`, domyślnie brany jest pierwszy serwer z pliku *~/.nwclient* i jest traktowany jako serwer preferowany. Na pierwszej pozycji w pliku powinieneś więc umieścić ten serwer plików, którego wolumeny montujesz najczęściej.

Bardziej skomplikowany przykład ncpmount

Przyjrzyjmy się bardziej skomplikowanemu przykładowi *ncpmount* wykorzystującemu właśnie opisane funkcje. Po pierwsze, napiszmy prosty plik *~/.nwclient*:

```
# Szczegóły logowania do serwera NetWare dla wirtualnego
# browaru i winiarni
#
# Konto browaru
ALES_F1/MATT staoicl
#
# Konto winiarni
REDS01/MATT staoicl
#

Aby upewnić się, że prawa dostępu są poprawne, wykonaj:

# chmod 600 ~/.nwclient
```

Zamontujmy jeden wolumen z serwera winiarni we współdzielonym podkatalogu, podając prawa dostępu do pliku i katalogu, tak by inni mogli korzystać z tych danych:

```
$ ncpmount -S REDS01 -V RESEARCH -f 0664 -d 0775 /usr/share/winery/data/
```

To polecenie w połączeniu z pokazanym plikiem *~/.nwclient* montuje wolumen `RESEARCH` z serwera `REDS01` w katalogu */usr/share/winery/data/*, używając konta `MATT` i pobierając hasło z pliku *~/.nwclient*. Prawa dostępu do zamontowanych plików to `0664`, a do katalogów – `0775`.

Kilka innych narzędzi IPX

Pakiet *ncpfs* zawiera szereg przydatnych narzędzi, których do tej pory nie opisaliśmy. Wiele z nich naśladuje narzędzia dostarczane z systemem NetWare. W tym podrozdziale przyjrzymy się najbardziej przydatnym z nich.

Lista serwerów

Polecenie *slist* pokazuje wszystkie serwery plików dostępne dla hosta. Informacje są w rzeczywistości pobierane z najbliższego rutera IPX. W założeniu polecenie miało za pewne pokazać użytkownikom, z jakich serwerów można montować wolumeny. Stało się jednak przydatnym narzędziem diagnostycznym, które pozwala administratorom sieci obserwować, jak są rozgłaszane informacje SAP:

```
$ slist
NPPWR-31-CD01          23A91330   000000000001
V242X-14-F02           A3062DB0   000000000001
QITG_284ELI05_F4       78A20430   000000000001
QRWMA-04-F16           B2030D6A   000000000001
VWPDE-02-F08           35540430   000000000001
NMCS_33PARK08_F2       248B0530   000000000001
NCCRD-00-CD01          21790430   000000000001
NWGNG-F07              53171D02   000000000001
QCON_7TOMLI04_F7       72760630   000000000001
W639W-F04              D1014D0E   000000000001
QCON_481GYM0G_F1       77690130   000000000001
VITG_SOE-MAIL_F4R      33200C30   000000000001
```

slist nie przyjmuje żadnych argumentów. Wynik pokazuje nazwę serwera plików, adres sieci IPX i adres hosta.

Wysyłanie komunikatów do użytkowników NetWare

NetWare obsługuje mechanizm wysyłania komunikatów do zalogowanych użytkowników. Polecenie *nsend* implementuje tę funkcję w Linuksie. Aby wysłać komunikat, musisz być zalogowany do serwera, a więc musisz podać w wierszu poleceń nazwę serwera, szczegóły dotyczące swojego konta, docelowego użytkownika oraz komunikat do wysłania:

```
# nsend -S vbrew_f1 -U gary -P j0yj0y supervisor "Chodźmy na piwo zanim zro-
bimy kolejki drukowania!"
```

Użytkownik o nazwie `gary` wysyła tu kuszące zaproszenie do osoby używającej konta `supervisor` na serwerze `ALES_F1`. Jeżeli nie podamy szczegółów, zostanie wykorzystany nasz domyślny serwer plików i dane o koncie.

Przeglądanie i operowanie danymi bindery

Każdy serwer plików NetWare posiada bazę informacji o użytkownikach i konfiguracji. Ta baza danych nosi nazwę *bindery*. Linux posiada zestaw narzędzi do jej odczytywania, a jeżeli masz prawa użytkownika **supervisor** na serwerze NetWare, pozwala także na wstawianie i usuwanie jej elementów. W tabeli 15-3 podajemy listę tych narzędzi.

Tabela 15-3. Narzędzia linuksowe do operacji na bindery

Nazwa polecenia	*Opis polecenia*
nwfstime	Wyświetla lub ustawia czas i datę serwera NetWare.
nwuserlist	Pokazuje użytkowników zalogowanych do serwera NetWare.
nwvolinfo	Wyświetla informacje o wolumenach NetWare.
nwbocreate	Tworzy obiekt bindery NetWare.
nwbols	Listuje obiekty bindery NetWare.
nwboprops	Pokazuje własności obiektu bindery NetWare.
nwborm	Usuwa obiekt bindery NetWare.
nwbpcreate	Tworzy własność obiektu bindery NetWare.
nwpbvalues	Drukuje zawartość własności obiektu bindery NetWare.
nwbpadd	Ustawia wartość własności obiektu bindery NetWare.
nwbprm	Usuwa własność bindery NetWare.

Drukowanie do kolejki NetWare

Pakiet *npcfs* zawiera małe narzędzie o nazwie *nprint*, które wysyła zadania do kolejki drukowania przez połączenie NCP NetWare. To polecenie tworzy połączenie, o ile takie nie istnieje, i wykorzystuje plik *~/.nwclient*, opisany wcześniej, by ukryć nazwę użytkownika i hasło przed wścibskimi osobami. Argumenty wiersza poleceń używane w procesie logowania są takie same jak argumenty polecenia *ncpmount*, a więc nie będziemy ich tu powtarzać. W naszym przykładzie omówimy najważniejsze opcje wiersza poleceń; szczegóły znajdziesz na stronach podręcznika elektronicznego *nprint(1)*.

Jedyną wymaganą opcją *nprint* jest nazwa pliku do wydrukowania. Jeżeli nazwa pliku zostanie podana w postaci znaku – lub nie zostanie w ogóle podana, *nprint* przyjmie zadanie ze standardowego wejścia. Najważniejsze opcje *nprint* to serwer plików i kolejka drukowania, do których chcesz wysłać zadanie. Tabela 15-4 zawiera najważniejsze opcje.

Tabela 15-4. Opcje wiersza poleceń nprint

Opcja	*Opis*
`-S` *`nazwa_serwera`*	Nazwa serwera plików NetWare obsługującego kolejkę drukowania, do której chcesz wysłać zadanie. Zwykle wygodnie jest mieć wpis dotyczący serwera w pliku *~/.nwclient*. Ta opcja jest obowiązkowa.
`-q` *`nazwa_kolejki`*	Kolejka drukowania, do której chcesz wysłać zadanie. Ta opcja jest obowiązkowa.
`-d` *`opis_zadania`*	Tekst, który pojawi się na konsoli drukowania na liście zakolejkowanych zadań.
`-l` *`wiersze`*	Liczba wierszy do wydrukowania na stronie. Domyślnie 66.
`-r` *`kolumny`*	Liczba kolumn do wydrukowania na stronie. Domyślnie 80.
`-c` *`kopie`*	Liczba kopii do wydrukowania. Domyślnie 1.

Prosty przykład wykorzystujący polecenie *nprint* wygląda tak:

```
$ nprint -S REDS01 -q PSLASER -c 2 /home/matt/ethylene.ps
```

To polecenie wydrukowałoby dwie kopie pliku */home/matt/ethylene.ps* na drukarce `PSLASER` podłączonej do serwera `REDS01` korzystając z konta (nazwy użytkownika i hasła) uzyskanego z pliku *~/.nwclient*.

Używanie nprint z demonem drukarki wierszowej

Przy pomnij sobie, jak mówiliśmy, że opcja *–c* polecenia *ncpmount* jest przydatna do drukowania. Teraz wyjaśnimy dlaczego.

Linux zwykle używa oprogramowania drukarki wierszowej w stylu BSD. Demon drukarki wierszowej (*lpd*) sprawdza lokalny katalog bufora, szukając w nim skolejkowanych zadań do wydrukowania. *lpd* odczytuje nazwę drukarki i inne parametry ze specjalnego pliku bufora i zapisuje dane na drukarkę, opcjonalnie przesyłając je przez filtr w celu zmiany lub wykonania na nich jakichś operacji.

Demon *lpd* wykorzystuje prostą bazę danych */etc/printcap*, w której zapisane są informacje konfiguracyjne, również o tym, jakie filtry uruchomić. *lpd* zwykle działa z prawami dostępu specjalnego użytkownika systemowego **lp**.

nprint możesz skonfigurować jako filtr dla *lpd*, co pozwoli użytkownikom Linuksa na wysyłanie danych bezpośrednio na drukarki zdalne, które są obsługiwane przez serwery plików NetWare. Aby to zrobić, użytkownik **lp** musi mieć możliwość wysłania żądań NCP do serwera NetWare.

Prostym sposobem na zrobienie tego bez potrzeby tworzenia przez **lp** własnego połączenia i logowania się do serwera jest określenie **lp** jako właściciela połączenia ustalonego przez innego użytkownika. Pełny przykład, jak skonfigurować system drukowania Linuksa do obsługi zadań drukowania od klientów Netware, składa się z trzech etapów:

1. Tworzenie skryptu pośredniego.

   Plik */etc/printcap* nie pozwala na podawanie filtrom opcji. Dlatego musisz napisać prosty skrypt, który wywoła polecenie wraz z opcjami. Skrypt pośredni może wyglądać tak:

   ```
   #!/bin/sh
   # p2pslaser - prosty skrypt przekierowujący stdin do kolejki
   # PSLASER na serwerze REDS01
   #
   /usr/bin/nprint -S REDS01 -U stuart -q PSLASER
   #
   ```

   Zapisz skrypt w pliku */usr/local/bin/p2pslaser*.

2. Umieszczenie wpisu w pliku */etc/printcap*.

   Będziemy musieli skonfigurować utworzony skrypt *p2pslaser* jako filtr wyjściowy w pliku */etc/printcap*. Będzie to wyglądało tak:

   ```
   pslaser|Postscript Laser Printer hosted by NetWare server:\
   :lp=/dev/null:\
   :sd=/var/spool/lpd/pslaser:\
   :if=/usr/local/bin/p2pslaser:\
   :af=/var/log/lp-acct:\
   :lf=/var/log/lp-errs:\
   :pl#66:\
   :pw#80:\
   :pc#150:\
   :mx#0:\
   :sh:
   ```

3. Dodanie opcji *–c* do polecenia *ncpmount*.

   ```
   ncpmount -S REDS01 .... -c lp ....
   ```

   Lokalny użytkownik **stuart** musi podać użytkownika **lp** jako właściciela połączenia, gdy zamontuje zdalny serwer NetWare.

Teraz dowolny użytkownik Linuksa może podać `pslaser` jako nazwę drukarki przy wywołaniu **lp**. Zadanie drukowania zostanie wysłane do określonego serwera NetWare i umieszczone w buforze do drukowania.

Zarządzanie kolejkami drukowania

Polecenie *pqlist* pokazuje wszystkie dostępne dla ciebie kolejki drukowania na danym serwerze. Jeżeli nie podasz serwera w wierszu poleceń, używając opcji *–S*, albo nie podasz nazwy użytkownika czy hasła, zostaną przyjęte domyślnie wpisy z pliku *~/.nwclient*:

```
# pqlist -S vbrew_f1 -U quest -n
Server: ALES_F1
Print queue name              Queue ID
-----------------------------------------------
TEST                          AA02009E
Q2                            EF0200D9
NPI223761_P1                  DA03007C
Q1                            F1060004
I-DATA                        0D0A003B
NPI223761_P3                  D80A0031
```

Nasz przykład pokazuje listę kolejek drukowania dostępnych dla użytkownika `guest` na serwerze `ALES_F1`*.

Aby obejrzeć zadania drukowania w kolejce, użyj polecenia *pqstat*. Jako argument przyjmuje ono nazwę kolejki i pokazuje wszystkie znajdujące się w niej zadania. Możesz opcjonalnie podać inny argument mówiący, ile zadań z kolejki chcesz zobaczyć. Poniższy przykład owy wynik został nieco zmniejszony, aby zmieścił się na stronie w tej książce:

```
$ pqsstat -S ALES_F1 NPI223761_P1

Server: ALES_F1   Queue: NPI223761_P1      Queue ID: 6A0E000C
  Seq   Name      Description              Status   Form Job ID
--------------------------------------------------------------
    1  TOTRAN   LyX document - proposal.lyx Active   0 02660001
```

Widzimy, że w kolejce czeka jedno zadanie będące własnością użytkownika `TOTRAN`. Pozostałe opcje zawarte w opisie zadania to jego status i identyfikator.

Polecenie *pqrm* jest używane do usuwania zadań ze wskazanej kolejki drukowania. Aby usunąć zadanie z kolejki, wydaj polecenie:

```
$ pqrm -S ALES_F1 NPI223761_P1 02660001
```

Polecenie jest proste, ale trudno go użyć z marszu. Warto poświęcić chwilę i przygotować sobie skrypt, który to ułatwi.

Emulacja serwera NetWare

Istnieją dwa bezpłatne emulatory serwerów plików NetWare dla Linuksa. Są to: *lwared*, opracowany przez Alesa Dryaka, i *mars_nwe*, opracowany przez Martina Stovera. Oba pakiety dają podstawową emulację serwera plików NetWare w Linuksie, umożliwiając klientom NetWare montowanie katalogów Linuksa wyeksportowanych jako wolumeny NetWare. Choć serwer *lwared* jest łatwiej skonfigurować, *mars_nwe* oferuje więcej funkcji. Instalacja i konfiguracja tych pakietów wykracza poza ramy tego rozdziału, ale oba są opisane w dokumencie *IPX-HOWTO*.

---

* Wygląda na to, że administratorzy systemu próbowali kilku produktów wirtualnego browaru przed ustaleniem nazw kolejek drukowania. Mamy nadzieję, że nazwy twoich kolejek są bardziej sensowne!

16

Zarządzanie UUCP Taylora



Pro to kół UUCP zo stał opra co wa ny pod ko niec lat sie dem dzie siątych przez Mike'a Le ska w AT&T Bell La bo ra to ries. Je go za da niem jest za pew nie nie pro stej sie ci komutowanej przez publiczne linie telefoniczne. Mimo popularności połączeń PPP i SLIP do In ter ne tu, wie le osób, któ re chcą mieć pocz tę elek tro niczną i gru py dys ku syjne Usene tu na swo ich do mo wych kom pu te rach, wciąż uży wa UUCP. Po pro stu jest to ta ń sze roz wią za nie, szcze gól nie w kra jach, gdzie użyt kow ni cy In ter ne tu muszą płacić za ka żdą mi nu tę miej sco wej roz mo wy te le fo nicz nej lub tam, gdzie nie mają lokal ne go do staw cy In ter ne tu i muszą płacić za roz mo wy za miej sco we. Choć ist nie je wie le im ple men ta cji UUCP działających na wie lu róż nych plat for mach sprz ę to wych i sys te mach ope ra cyj nych, są one ze sobą kom pa ty bil ne.

Jed nak tak jak z wię k szo ścią opro gra mo wa nia, któ re w ja kiś spo sób stało się przez la ta „stan dar dem", nie ma UUCP, któ re na zy wałoby się po pro stu UUCP. Od za implemen to wa nia pierw szej wer sji w 1976 ro ku przeszło ono pewną ewo lu cję. Obec nie ist nieją dwie głów ne od mia ny, róż niące się przede wszyst kim wspar ciem sprz ę towym i spo so bem kon fi gu ra cji. Mają one własne im ple men ta cje, a ka żda z nich róż ni się od po zo stałych w bar dzo nie wiel kim stop niu.

Jed na od mia na jest zna na ja ko 2. wer sja UUCP i jej hi sto ria się ga im ple men ta cji Mike'a Le ska, Davi da A. Novit za i Gre ga Ches so na z ro ku 1977. Mi mo swo ich lat wciąż jest czę sto uży wa na. No we im ple men ta cje tej wer sji są na praw dę bar dziej funk cjonal ne niż now sze od mia ny UUCP.

Dru ga od miana zo stała oprac owa na w 1983 ro ku i jest po wszechn ie na zyw ana BNU (*Basic Networking Utilities*) lub Ho neyD anBer UUCP. Ta ostatn ia na zwa po chod zi od na zwisk au torów (P. Ho neym an, D.A. Novitz i B. E. Red man) i czę sto jest skra ca na do postaci HDB; tego okre śle nia będzie my używ ali w tym roz dziale. HDB miała usunąć pew ne bra ki 2. wersji UUCP. Na przykład zo stały do dane no we pro tokoły trans mis ji, a ka tal og bu fora zo stał po dziel ony tak, że te raz jest je den ws pó lny katalog dla wszyst kich ośrodk ów dla których obsługuj esz ruch UUCP.

Implementacja UUCP dystrybuowana obecnie z Linuksem to tak zwane UUCP Taylora wersja 1.06 i o niej traktuje niniejszy rozdział*. Pakiet Taylora został wydany w sierpniu 1995 roku. Poza pracą z tradycyjnymi plikami konfiguracyjnymi może być także skompilowany tak, by rozumieć pliki konfiguracyjne nowego typu – znane również pod nazwą Taylor.

UUCP Taylora jest zwykle kompilowane do wersji kompatybilnej z HDB, schematem konfiguracyjnym Taylora lub obydwoma. Ponieważ schemat Taylora jest bardziej elastyczny niż niejasne pliki konfiguracyjne HDB, opiszemy poniżej właśnie ten schemat.

Ten rozdział nie jest pomyślany jako wyczerpujący opis opcji wiersza poleceń UUCP, ale jako wprowadzenie do skonfigurowania działającego węzła UUCP. Pierwszy podrozdział informuje o tym, jak UUCP implementuje zdalne wykonywanie poleceń i przesyłanie plików. Jeżeli nie jesteś zupełnym nowicjuszem w branży UUCP, możesz go pominąć i przejść do dalszego podrozdziału *Pliki konfiguracyjne UUCP*, który wyjaśnia, jak różne pliki są wykorzystane do konfiguracji UUCP.

Zakładamy jednak, że znasz programy użytkownika pakietu UUCP, czyli *uucp* i *uux*. Ewentualnie ich opis znajdziesz na stronach podręcznika elektronicznego.

Poza publicznie dostępnymi programami *uucp* i *uux*, pakiet UUCP zawiera szereg poleceń używanych jedynie do celów administracyjnych. Służą one do monitorowania ruchu UUCP twojego węzła, usuwania starych plików log czy kompilowania statystyk. Nie będziemy ich tu taj opisywać, ponieważ wykonują zadania dodatkowe. Poza tym są doskonale udokumentowane i łatwe w obsłudze. Więcej informacji znajdziesz na stronach podręcznika elektronicznego. Jednak istnieje trzecia kategoria programów: te, które „odwalają całą czarną robotę” UUCP. Są to *uucico* (gdzie *cico* pochodzi od słów *copy-in copy-out*) i *uuxqt*, które wykonuje zadania przysłane przez systemy zdalne. W tym rozdziale skoncentrujemy się na tych dwóch istotnych programach.

Jeżeli nie jesteś zadowolony z naszego wyboru tematów, powinieneś przeczytać dokumentację dostarczaną wraz z pakietem UUCP. Jest to zestaw plików Texinfo, które opisują konfigurację z wykorzystaniem schematu Taylora. Pliki Texinfo możesz przekształcić w plik *dvi* za pomocą *texi2dvi* (który można znaleźć w pakiecie Texinfo w twojej dystrybucji) i obejrzeć go, używając polecenia *xdvi*.

Kolejnym doskonałym źródłem informacji na temat UUCP w środowisku Linuksa jest *UUCP-HOWTO* autorstwa Guylhema Aznara. Jest ono dostępne w ramach Projektu Dokumentacji Linuksa i regularnie wysyłane do grupy *comp.os.linux.answers*.

Istnieje również grupa dyskusyjna poruszająca tematy związane z UUCP: *comp.mail.uucp*. Jeżeli masz pytania szczegółowe dotyczące UUCP Taylora, lepiej je zadać właśnie tu, a nie w grupach z serii *comp.os.linux.**.

---

* Napisana i zastrzeżona przez Iana Taylora w 1995 roku.

Przesyłanie i zdalne wykonywanie w UUCP

Dla zrozumienia UUCP istotne jest pojęcie zadań. Każda transmisja zainicjowana przez użytkownika za pomocą *uucp* lub *unx* nazywa się *zadaniem*. Składa się ono z poleceń do wykonania na hoście zdalnym, zestawu plików do przesłania między ośrodkami lub obu tych elementów.

Jako przykład weźmy poniższe polecenie, które kopiuje przez UUCP plik *netguide.ps* do zdalnego hosta **pablo** i wykonuje na nim polecenie *lpr* drukujące plik:

```
$ uux -r pablo!lpr !netguide.ps
```

Generalnie UUCP nie wywołuje natychmiast zdalnego hosta, by wykonać zadanie (co mógłbyś zrobić za pomocą *kermit*). Sporządza natomiast tymczasowy opis zadania. Nazywa się to *buforowaniem* (ang. *spooling*). Drzewo katalogów, w którym są umieszczane zadania, nazywa się *katalogiem buforowym* i przeważnie znajduje się w katalogu */var/spool/uucp*. W naszym przykładzie opis zadania będzie zawierał informacje o zdalnym poleceniu do wykonania (*lpr*), użytkowniku, który zlecił jego wykonanie, i kilku innych elementach. Poza opisem zadania, UUCP musi zachować plik wejściowy *netguide.ps*.

Dokładna lokalizacja i nazewnictwo plików buforowych może się różnić w zależności od opcji wybranych w czasie kompilacji. UUCP kompatybilne z HDB zwykle zachowuje pliki buforowe w katalogu */var/spool/uucp* w podkatalogu o nazwie ośrodka zdalnego. W przypadku kompilacji z konfiguracją Taylora, UUCP tworzy w tym katalogu podkatalogi dla różnych typów plików buforowych.

W regularnych odstępach czasu UUCP dzwoni do zdalnego systemu. Gdy zostanie nawiązane połączenie z systemem, UUCP przesyła pliki opisujące zadanie oraz wszelkie pliki wejściowe. Przychodzące zadanie nie zostanie wykonane natychmiast, ale po zakończeniu połączenia. Wykonanie jest obsługiwane przez *uuxqt*, który także obsługuje przekazywanie wszelkich zadań przeznaczonych dla drugiego ośrodka.

Aby rozróżnić mniej i bardziej istotne zadania, UUCP każdemu z nich nadaje *stopień* (ang. *grade*). Jest to cyfra z przedziału od 0 do 9, litera z przedziału od A do Z oraz od a do z w kolejności malejącej priorytetów. Poczta jest zwykle buforowana ze stopniem B lub C, natomiast grupy dyskusyjne ze stopniem N. Zadania o wyższych stopniach są przesyłane w pierwszej kolejności. Stopnie mogą być przypisywane za pomocą opcji *–g* w wywołaniu *uucp* lub *uux*.

W pewnych okresach czasu możesz również zabronić przesyłania zadań o stopniu mniejszym niż zadany. Aby to zrobić, ustaw *maksymalny stopień buforowania* (ang. *maximum spool grade*), który będzie dopuszczalny w czasie konwersacji. Maksymalny stopień buforowania domyślnie ma wartość *z*, co oznacza, że zadania o wszystkich stopniach będą przesyłane za każdym razem. Zauważ, że składnia jest tu nieco niejasna: plik jest przesyłany jedynie wtedy, gdy ma stopień *równy* lub *większy* niż maksymalny próg buforowania.

Wewnętrzne działanie uucico

Krótki opis tego, jak w rzeczywistości następuje połączenie ze zdalnym systemem, pomoże zrozumieć, dla czego *uucico* musi znać pewne informacje.

Gdy uruchomisz *uucico* `–s system` z wiersza poleceń, *uucico* najpierw musi zrealizować połączenie fizyczne. Podejmowane działania zależą od rodzaju połączenia, jakie ma być otwarte. W przypadku linii telefonicznej wymaga to znalezienia modemu i zadzwonienia. W przypadku TCP, *uucico* musi wywołać *gethostbyname*, aby zamienić nazwę na adres sieci, stwierdzić, który port otworzyć, i powiązać adres z odpowiednim gniazdem.

Po poprawnym nawiązaniu połączenia następuje uwierzytelnienie. Ta procedura ogólnie składa się z zapytania zdalnego systemu o nazwę użytkownika i ewentualnie hasło. Wymiana tych danych jest powszechnie nazywana *dialogiem logowania* (ang. *login chat*). Procedura uwierzytelniania jest wykonywana albo przez typowy zestaw *getty/login*, albo przez sam *uucico* na gniazdach TCP. Jeżeli uwierzytelnienie powiedzie się, druga strona uruchamia *uucico*. Kopia *uucico* po stronie, która zainicjowała połączenie, czyli lokalnej, jest nazywana *nadrzędną* (ang. *master*), a zdalna kopia jest nazywana *podległą* (ang. *slave*).

Później następuje *faza uzgadniania* (ang. *handshake phase*): system nadrzędny wysyła swoją nazwę hosta i kilka znaczników. System podległy sprawdza tę nazwę hosta pod względem prawa logowania, wysyłania i odbioru plików itd. Znaczniki opisują (między innymi) maksymalny stopień plików buforowych, pozwalający na ich przesłanie. Jeżeli jest włączony licznik konwersacji lub *numer kolejny wywołania* (ang. *call sequence number*), to są teraz sprawdzane. Dzięki tej funkcji obie strony mogą posiadać licznik poprawnych połączeń i je porównywać. Jeżeli liczniki się nie zgadzają, uzgadnianie się nie udaje. Jest to przydatne do zabezpieczenia się przed oszustami.

Na koniec oba *uucico* próbują uzgodnić wspólny *protokół transmisji*. Protokół ten decyduje o sposobie przesyłania danych, sprawdzaniu ich spójności i retransmisji w przypadku błędów. Potrzebne są różne protokoły, ponieważ obsługiwane są różne typy połączeń. Na przykład linie telefoniczne wymagają „bezpiecznego" protokołu, który jest nieufny i wszędzie węszy błędy, natomiast transmisja TCP jest z założenia niezawodna i może używać efektywniejszego protokołu, który nie wykonuje dodatkowego sprawdzania błędów.

Po zakończeniu uzgadniania rozpoczyna się faza rzeczywistej transmisji. Obie strony włączają wybrany sterownik protokołu. W tym miejscu sterowniki wykonują sekwencję inicjacyjną specyficzną dla protokołu.

Następnie system nadrzędny wysyła wszystkie skolejkowane pliki do hosta zdalnego, którego stopień buforowania jest wystarczająco wysoki. Gdy skończy, informuje system podrzędny, że zrobił swoje i że ten może się rozłączyć. W tym momencie system podrzędny może zgodzić się na rozłączenie lub przejąć konwersację. Następuje zamiana ról: system zdalny staje się nadrzędnym, a lokalny staje się podległym. Nowy system nadrzędny wysyła swoje pliki. Gdy zakończy, oba *uucico* wymieniają komunikaty zakończenia i zamykają połączenie.

Jeżeli potrzebujesz dodatkowych informacji na temat UUCP, zajrzyj do kodu źródłowego. Po sieci krąży również napraw dę zabytkowy artykuł, napisany przez Davida A. Novitza, ze szczegółowym opisem protokołu UUCP*. Lista pytań FAQ *Taylor UUCP* również omawia pewne szczegóły implementacji UUCP. Dokument ten jest regularnie wysyłany do grupy dyskusyjnej *comp.mail.uucp*.

Opcje wiersza poleceń uucico

Tutaj podamy najważniejsze opcje wiersza poleceń *uucico*:

--system, *-s* `system`
: Dzwonienie do zadanego `systemu`, dopóki nie zostanie to zabronione przez ograniczenia czasowe.

-S `system`
: Dzwonienie do `systemu` bezwarunkowo.

-- master, *-r1*
: Uruchomienie *uucico* w trybie nadrzędnym. Jest to opcja domyślna, jeżeli zostanie podane *–s* lub *–S*. Opcja *–r1* powoduje, że *uucico* próbuje dzwonić do wszystkich systemów umieszczonych w pliku *sys*, opisanym w następnym podrozdziale, dopóki nie upłynie czas przewidziany na rozmowę lub powtórne dzwonienie.

--slave, *-r0*
: Uruchomienie *uucico* w trybie podległym. Jest to tryb domyślny, jeżeli nie zostaną podane opcje *–s* lub *–S*. W trybie tym zakłada się, że używane jest standardowe wejście/wyjście podłączone do portu szeregowego albo do portu TCP określonego przez opcję *–p*.

--ifwork, *–C*
: Ta opcja uzupełnia *–s* lub *–S* mówiąc programowi *uucico*, by dzwonił do wymienionych systemów tylko wtedy, gdy w buforze są dla nich zadania.

-- debug `typ`, *-x* `typ`, *-X* `typ`
: Włączenie debugowania zadanego typu. Może być podane kilka typów w postaci listy oddzielonej przecinkami. Dopuszczalne są następujące typy: `abnormal`, `chat`, `handshake`, `uucp-proto`, `proto`, `port`, `config`, `spooldir`, `execute`, `incoming` i `outgoing`. Użycie `all` włącza wszystkie opcje. Dla kompatybilności z implementacjami UUCP można podać także liczbę, która włącza debugowanie dla `n` pierwszych elementów z powyższej listy.

Komunikaty z debugowania zostaną zapisane do pliku *Debug* w katalogu */var/spool/uucp*.

Pliki konfiguracyjne UUCP

W odróżnieniu od prostszych programów do przesyłania plików, UUCP zostało zaprojektowane tak, by obsługiwało automatycznie wszystkie transmisje. Jeśli zostanie poprawnie skonfigurowane, to nie wymaga stałej opieki administratora. Infor-

* Zawiera go także książka *System Manager's Manual* dołączona do systemu 4.4BSD

macje potrzebne do automatycznej transmisji są przechowywane w kilku plikach konfiguracyjnych, znajdujących się w katalogu */usr/lib/uucp*. Większość z nich jest używana tylko w czasie dzwonienia.

Łagodne wprowadzenie do UUCP Taylora

Powiedzenie, że konfiguracja UUCP jest trudna, będzie niedomówieniem. W rzeczywistości jest bardzo skomplikowana i czasem nawet zwięzły format plików konfiguracyjnych nie ułatwia sprawy (choć format Taylora i tak czyta się łatwo w porównaniu ze starszymi formatami HDB lub wersji 2).

Aby pokazać, jak współdziałają wszystkie pliki konfiguracyjne, przedstawimy najważniejsze z nich i przyjrzymy się prostym wpisom w tych plikach. Nie będziemy teraz wyjaśniać wszystkiego szczegółowo. Dokładniejsze informacje są podane w następnych podrozdziałach. Gdybyś chciał skonfigurować UUCP na swoim komputerze, najlepiej zacząć od przykładowych plików i po kolei je adaptować do własnych potrzeb. Możesz wykorzystać tu pokazane pliki lub te załączone w twojej ulubionej dystrybucji Linuksa.

Wszystkie pliki opisane w tym podrozdziale znajdują się w katalogu */etc/uucp* lub jego podkatalogach. Dystrybucje Linuksa zawierają binaria UUCP, które obsługują zarówno konfigurację HDB, jak i schemat Taylora, ale każdy ze stawów plików ma swój własny podkatalog. W katalogu */usr/lib/uucp* zwykle będzie znajdował się plik *README*.

Aby UUCP działało poprawnie, pliki te muszą należeć do użytkownika **uucp**. Niektóre z nich zawierają hasła i numery telefonów i dlatego powinny mieć prawo dostępu 600. Zauważ, że choć większość poleceń UUCP musi mieć prawo setuid użytkownika **uucp**, musisz pamiętać, że nigdy *nie może* go mieć program *uuchk*. W przeciwnym razie użytkownicy będą mogli wyświetlać hasła systemowe, nawet jeżeli pliki z hasłami będą miały prawo do stępu ustawione na 600.

Głównym plikiem konfiguracyjnym UUCP jest */etc/uucp/config*, w którym są ustawiane parametry ogólne. Najważniejszy z nich (i w tej chwili jedyny) to nazwa tego hosta UUCP. W wirtualnym browarze gate way em UUCP jest host **vstout**.

```
# /etc/uucp/config - g□ówny plik konfiguracyjny UUCP
nodename      vstout
```

Plik *sys* jest kolejnym ważnym plikiem konfiguracyjnym. Zawiera wszystkie informacje specyficzne dla systemów, z którymi jesteś połączony. Należą do nich nazwa ośrodka i informacje o samym łączu, takie jak numer telefonu, jeżeli jest wykorzystywane łącze modemowe. Typowy wpis dla ośrodka **pablo** podłączonego przez modem wyglądałby tak:

```
# /usr/lib/uucp/sys - nazwy s□siadów UUCP
# system: pablo
system    pablo
time      Any
phone     555-22112
port      serial1
speed     38400
chat      ogin: vstout ssword: lorca
```

`time` określa, o której godzinie system zdalny może być wywoływany. `chat` opisuje skrypty dialogu logowania – kolejne ciągi, które muszą być wymienione, aby *uucico* mogło zalogować się do **pablo**. Do skryptów dialogu logowania jeszcze wrócimy. Słowo kluczowe `port` nadaje po prostu nazwę wpisowi w pliku *port* (patrz rysunek 16-1). Możesz przypisać dowolną nazwę, o ile odwołuje się do poprawnego wpisu w pliku *port*.

Plik *port* zawiera informacje specyficzne dla samego łącza. Dla łączy modemowych opisuje specjalny plik urządzenia, jaki ma być użyty, zakres obsługiwanych prędkości i typ urządzenia podłączonego do portu. Poniższy wpis opisuje */dev/ttyS1* (czyli COM2), do którego administrator podłączył modem NakWell, który może działać z prędkością do 38400 bitów na sekundę. Nazwa portu jest dobrana tak, by odpowiadała tej z pliku *sys*:

```
# /etc/uucp/port - porty UUCP
# /dev/ttyS1 (COM2)
port      serial1
type      modem
device    /dev/ttyS1
speed     38400
dialer    nakwell
```

Informacja na temat modemów jest przechowywana w jeszcze innym pliku o nazwie *dial*. Dla każdego typu modemu zawiera on ciąg poleceń, które trzeba wykonać, aby połączyć się z ośrodkiem zdalnym o zadanym numerze telefonicznym. Znów jest to określone przez skrypt dialogowy. Na przykład wpis dla NakWell mógłby wyglądać tak:

```
# /etc/uucp/dial - informacje o dzwoni□cych
# modemy NakWell
dialer      nakwell
chat          * * AT&F OK ATDT\T CONNECT
```

Wiersz rozpoczynający się od `chat` określa dialog modemu, czyli ciąg poleceń wysyłanych i odbieranych przez modem w celu jego inicjacji i wykonania połączenia z żądanym numerem. Sekwencja `\T` zostanie zastąpiona przez *uucico* numerem telefonu.

Abyś z grubsza miał pojęcie, jak *uucico* wykorzystuje te pliki konfiguracyjne, załóżmy, że wydajesz następujące polecenie:

```
$ uucico -s pablo
```

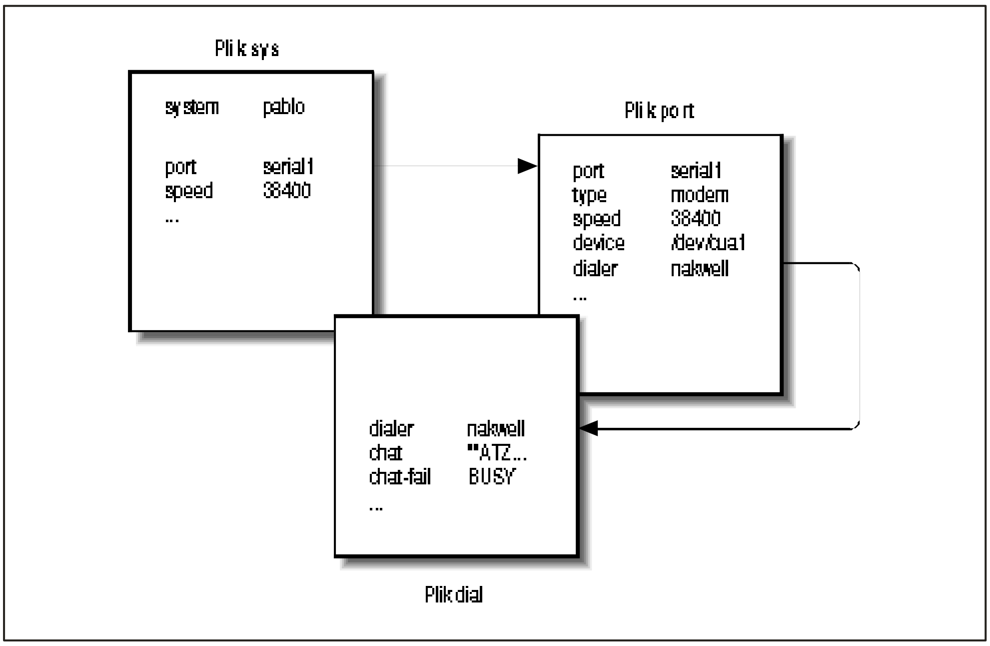
Pierwsza rzecz, jaką robi *uucico*, to poszukanie **pablo** w pliku *sys*. Na podstawie wpisu **pablo** w pliku *sys* wiadomo, że połączenie należy zrealizować przez port `serial1`. Plik *port* mówi *uucico*, że jest to port modemu, do którego podłączony jest modem NakWell.

Następnie *uucico* poszukuje w pliku *dial* wpisu dla modemu NakWell. Po jego znalezieniu otwiera port szeregowy */dev/cua1* i wykonuje dialog dzwonienia. Oznacza to, że wysyła *AT&F*, czeka na odpowiedź *OK* itd. Gdy napotka ciąg `\T`, zastępuje go numerem telefonu (555-22112) uzyskanym z pliku *sys*.

Gdy modem zwróci ciąg *CONNECT*, połączenie zostaje nawiązane i dialog modemu zostaje zakończony. *uucico* powraca do pliku *sys* i wykonuje dialog logowania. W naszym przykładzie będzie on czekał na monit *login:*, a następnie wyśle nazwę użytkownika (**vstout**), poczeka na monit *password:* i wyśle hasło (*lorca*).

Zakłada się, że po zakończeniu procedury uwierzytelniania zdalny system uruchomi *uucico* po swojej stronie. Następnie obie strony przejdą do fazy uzgadniania, opisanej w poprzednim podrozdziale.

Rysunek 16-1 pokazuje zależności pomiędzy plikami konfiguracyjnymi.



Rysunek 16-1. Powiązania plików konfiguracyjnych UUCP Taylora

Co musi wiedzieć UUCP

Zanim zaczniesz tworzyć pliki konfiguracyjne UUCP, musisz wiedzieć co nieco o jego wymaganiach.

Najpierw musisz stwierdzić, do jakiego portu szeregowego podłączony jest twój modem. Zwykle porty (DOS-a) od COM1: do COM4: odzwierciedlają pliki specjalne */dev/ttS0* do */dev/ttyS3*. Niektóre dystrybucje, takie jak Slackware, tworzą dowiązanie */dev/modem* do odpowiedniego pliku urządzenia *ttyS** i konfigurują programy komunikacyjne, takie jak *kermit*, *seyon*, w ten sposób, by używały tego dowiązania. W takim przypadku powinieneś używać */dev/modem* także w swojej konfiguracji UUCP.

A oto dlaczego sięga się po dowiązanie symboliczne. Wszystkie programy dzwoniące używają tak zwanych *plików blokujących* do sygnalizowania, że port szeregowy

jest zajęty. Nazwy tych plików blokujących są połączeniem ciągu *LCK..* i nazwy pliku urządzenia, na przykład *LCK..ttyS1*. Jeżeli programy różnie nazywają to samo urządzenie, nie uda im się rozpoznać innych plików blokujących. W konsekwencji, będą zakłócały sobie wzajemnie sesje, jeżeli zostaną uruchomione w tym samym czasie. Jest to możliwe, gdy zaszeregujesz wywołania UUCP za pomocą wpisu w *crontab*. Szczegóły konfiguracji portu szeregowego znajdziesz w rozdziale 4, *Konfigurowanie urządzeń szeregowych*.

Następnie musisz stwierdzić, z jaką prędkością Linux komunikuje się z twoim modemem. Musisz ustawić tę prędkość na najbardziej efektywną wartość, jaką chciałbyś uzyskać. Efektywna przepustowość może być dużo wyższa niż surowa przepustowość fizyczna zapewniana przez modem. Na przykład wiele modemów wysyła i odbiera dane z prędkością 56 kbps. Przy zastosowaniu protokołów kompresji, takich jak V.42bis, rzeczywista przepustowość może przekroczyć 100 kbps.

Oczywiście jeżeli UUCP ma mieć niewiele do roboty, to wystarczy ci numer telefonu do systemu, do którego chcesz dzwonić. Potrzebujesz także poprawnej nazwy użytkownika i hasła na maszynie zdalnej*.

Musisz także *dokładnie* wiedzieć, jak zalogować się do systemu. Czy musisz nacisnąć klawisz [Enter], za nim pojawi się monit logowania? Czy ma on postać `login:` czy `user:`? Jest to wiedza niezbędna do stworzenia *skryptu dialogowego*. Jeżeli tego nie wiesz lub jeżeli typowe skrypty nie działają, spróbuj zadzwonić do systemu za pomocą programu terminala, na przykład *kermit* czy *minicom*, i zanotuj dokładnie to, co zobaczysz.

Nazewnictwo ośrodków

Tak jak w sieciach opartych na TCP/IP, twój host musi mieć nazwę w sieci UUCP. Dopóki chcesz wykorzystywać UUCP tylko to przesyłania plików w sieci lokalnej lub pomiędzy ośrodkami, do których dzwonisz bezpośrednio, nazwa ta nie musi być zgodna z żadnymi standardami**.

Jeżeli jednak używasz UUCP do połączenia z pocztą lub grupami dyskusyjnymi, powinieneś pomyśleć o nazwie zarejestrowanej w projekcie mapowania UUCP***. Projekt mapowania UUCP jest opisany w rozdziale 17, *Poczta elektroniczna*. Nawet jeżeli znajdujesz się w domenie, możesz rozważyć posiadanie oficjalnej nazwy UUCP dla twojego ośrodka.

---

* Jeżeli tylko testujesz UUCP, wykorzystaj numer pobliskiego ośrodka archiwum. Zapisz nazwę użytkownika i hasło (są one publicznie dostępne), by umożliwić anonimowe kopiowanie plików. W większości przypadków jest to coś w stylu **uucp/uucp** lub **nuucp/uucp**

** Jedynym ograniczeniem jest to, że nie powinna być dłuższa niż siedem znaków, a więc nie mieszaj ze sobą implementacji UUCP, które działają w systemach operacyjnych narzucających większe ograniczenia co do nazw plików. Nazwy, które są dłuższe niż siedem znaków, są często przez UUCP skracane. Niektóre wersje UUCP ograniczają nazwy nawet do sześciu znaków.

*** Projekt mapowania UUCP (*UUCP Mapping Project*) rejestruje wszystkie nazwy hostów UUCP z całego świata i zapewnia ich unikalność.

Często ludzie wybierają sobie nazwy UUCP tak, by pasowały do pierwszego elementu ich pełnej nazwy domenowej. Załóżmy, że adres domeny, w której jest twój ośrodek, to **swim.twobirds.com**, a więc nazwa twojego hosta UUCP będzie brzmiała **swim**. Traktuj ośrodki UUCP tak, jakby znały się po imieniu. Oczywiście możesz także użyć nazwy UUCP zupełnie niezwiązanej z twoją pełną nazwą domenową.

Jednak sprawdź, czy w adresach pocztowych przypadkiem nie posłużyłeś się niepełną nazwą ośrodka. Nie masz się czego obawiać, jeśli zarejestrowałeś ją jako swoją oficjalną nazwę. W najlepszym wypadku poczta do niezarejestrowanego hosta UUCP zniknie w jakiejś dużej czarnej dziurze. Jeżeli używasz nazwy wykorzystywanej już przez jakiś inny ośrodek, poczta będzie do niego przekierowana i przysporzy administratorowi poczty bólu głowy.

Domyślnie pakiet UUCP używa nazwy definiowanej przez *hostname* jako nazwy ośrodka UUCP. Ta nazwa jest przeważnie ustawiana w czasie inicjacji systemu w skryptach *rc* i zwykle znajduje się w pliku */etc/hostname*. Jeżeli twoja nazwa UUCP jest inna niż ta, która znajduje się w tym pliku, musisz użyć opcji *hostname* w pliku *config*, by poinformować *uucico*, jaka jest twoja nazwa UUCP. Opisujemy to dalej.

Pliki konfiguracyjne Taylora

Powróćmy teraz do plików konfiguracyjnych. UUCP Taylora pobiera swoje informacje z następujących plików:

config
: Jest to główny plik konfiguracyjny. Możesz w nim zdefiniować nazwę UUCP swojego ośrodka.

sys
: Ten plik opisuje wszystkie znane ośrodki. Dla każdego z nich posiada wpis z jego nazwą, godzinami wywołania, numerem, pod jaki należy dzwonić (o ile istnieje), typem urządzenia, którego trzeba używać, i sposobem logowania.

port
: Ten plik zawiera wpisy charakteryzujące każdy z dostępnych portów wraz z obsługiwaną prędkością i typem modemu.

dial
: Ten plik opisuje typy urządzeń używane do realizacji połączenia telefonicznego.

dialcode
: Ten plik zawiera rozwinięcia symbolicznych numerów kierunkowych.

call
: Ten plik zawiera nazwę użytkownika i hasło używane przy dzwonieniu do systemu. Jest rzadko używany.

passwd

Ten plik zawiera nazwy użytkowników i hasła, którymi systemy mogą posługiwać się przy logowaniu. Jest używany tylko wtedy, gdy *uucico* przeprowadza własne sprawdzanie haseł.

Pliki konfiguracyjne Taylora ogólnie składają się z wierszy zawierających pary słowo_kluczowe-wartość. Znak hasha oznacza wiersz z komentarzem. Aby użyć znaku # we własnej roli, zamaskuj go odwrotnym ukośnikiem, czyli tak: `\#`.

Istnieje szereg opcji, które możesz dostosować, używając tych plików konfiguracyjnych. Nie możemy przejrzeć wszystkich parametrów, ale omówimy tu najważniejsze z nich. Po ich omówieniu powinieneś być w stanie skonfigurować łącze UUCP oparte na modemie. Następne podrozdziały opisują modyfikacje wymagane do używania UUCP w sieci TCP/IP lub przez bezpośrednie łącze szeregowe. Pełny opis znajduje się w dokumencie Texinfo dołączonym do źródeł UUCP Taylora.

Gdy uznasz, że w pełni skonfigurowałeś system UUCP, możesz sprawdzić swoją konfigurację, używając narzędzia *uuchk* (znajdującego się w katalogu */usr/lib/uucp*). *uuchk* odczytuje twoje pliki konfiguracyjne i drukuje szczegółowy raport o konfiguracji używanej dla każdego z systemów.

Ogólne opcje konfiguracyjne używane w pliku config

W zasadzie tego pliku nie będziesz używał do niczego innego poza zdefiniowaniem nazwy hosta UUCP. Domyślnie UUCP będzie wykorzystywać nazwę ustaloną poleceniem *hostname*, ale dobrze jest jednak ustawić nazwę UUCP w sposób jawny. Oto przykład pliku *config*:

```
# /usr/lib/uucp/config - g□ówny plik konfiguracyjny UUCP
hostname       vstout
```

W tym pliku można ustawić także szereg innych parametrów, takich jak nazwa katalogu buforującego czy prawa dostępu dla anonimowego UUCP. Omówimy je w dalszej części tego rozdziału, w podrozdziale *Anonimowe UUCP*.

Jak za pomocą pliku sys powiedzieć UUCP o innych systemach

Plik *sys* opisuje systemy znane twojemu komputerowi. Dowolny wpis w tym pliku jest poprzedzany słowem kluczowym `system`, a kolejne wiersze, aż do następnego słowa `system`, określają szczegółowe parametry dla danego ośrodka. Przeważnie taki wpis definiuje parametry, takie jak numer telefonu i dialog logowania.

Parametry występujące przed pierwszym wpisem `system` są używane dla wszystkich systemów. Zwykle w sekcji parametrów domyślnych ustawiasz parametry protokołu i tym podobne.

Najczęściej spotykane pola są omówione szczegółowo dalej w tym rozdziale.

Nazwa systemu

Polecenie *system* określa nazwę systemu zdalnego. Musisz podać jego poprawną nazwę, a nie alias, ponieważ *uucico* sprawdza w czasie logowania, jaką nazwą przedstawia się system zdalny*.

Nazwa każdego systemu może pojawić się tylko raz. Gdybyś chciał użyć kilku zestawów konfiguracji dla tego samego systemu (na przykład różnych numerów telefonu, które *uucico* powinno pokolei wypróbowywać), możesz użyć słowa *alternatives*, które opiszemy po podstawowych opcjach konfiguracyjnych.

Numer telefonu

Jeżeli chcemy się łączyć z systemem zdalnym przez linię telefoniczną, pole `phone` opisuje numer, pod który należy dzwonić. Może zawierać kilka leksemów interpretowanych przez procedurę dzwonienia *uucico*. Znak równości (=) oznacza czekanie na powtórny ton, a znak minus (-) generuje jednosekundową przerwę. Niektóre instalacje telefoniczne nie chcą działać, jeżeli nie dodasz przerwy pomiędzy specjalnym kodem dostępu a numerem telefonu**.

Często wygodnie jest używać nazw zamiast numerów przy opisywaniu numerów kierunkowych. Plik *dialcode* pozwala powiązać nazwę z kodem, który można następnie wykorzystać przy wpisywaniu numeru telefonu do hostów zdalnych. Załóżmy, że masz następujący plik *dialcode*:

```
# /usr/lib/uucp/dialcode - t□umaczenie dialcode
Bogoham     024881
Coxton      035119
```

Mając takie tłumaczenie, możesz użyć w pliku *sys* numeru `Bogoham7732`, co prawdopodobnie spowoduje, że wszystko będzie bardziej czytelne i łatwiejsze do uaktualniania, gdyby numer kierunkowy do Bogoham kiedykolwiek się zmienił.

Port i prędkość

Opcje `port` i `speed` są wykorzystywane do wskazania urządzenia, przez które nawiązywane jest połączenie z systemem zdalnym, i prędkości, z jaką ma działać***. Wpis `system` może zawierać każdą z tych opcji oddzielnie lub obie razem. Przy poszukiwaniu odpowiedniego urządzenia w pliku *port*, wybierane są tylko porty o pasującej nazwie i/lub zakresie prędkości.

Generalnie użycie samej opcji `speed` powinno wystarczyć. Gdybyś w pliku *port* miał zdefiniowane tylko jedno urządzenie szeregowe, *uucico* zawsze wybierałoby poprawnie, a więc musisz mu podać tylko żądaną prędkość. Gdybyś do swojego komputera miał podłączone kilka modemów, wciąż nie warto nadawać nazwy każ-

---

* Starsze wersje 2 UUCP nie rozgłaszały swoich nazw przy połączeniu. Jednak nowsze implementacje często to robią, również UUCP Taylora.

** Na przykład większość wewnętrznych sieci telefonicznych w firmach wymaga, być dzwonił na zewnątrz przez 0 lub 9.

*** Szybkość transmisji *tty* w bitach na sekundę musi być przynajmniej tak duża, jak maksymalna przepustowość łącza.

demu portowi, ponieważ gdyby *uucico* stwierdziło, że kilka z nich pasuje, próbowałoby każde go urządzenia po kolei, aż natrafiłoby na nieużywane.

Dialog logowania

Spotkaliśmy się już ze skryptem logowania, który mówi *uucico*, jak zalogować się do systemu zdalnego. Składa się on z listy leksemów okreslających oczekiwane i wysyłane przez lokalny proces *uucico* ciągi znaków. *uucico* czeka aż zdalny host wyśle monit logowania, następnie podaje nazwę użytkownika, czeka na wysłanie przez system zdalny zapytania o hasło i wysyła hasło. Oczekiwane i wysyłane ciągi znaków są umieszczone w skrypcie na zmianę. *uucico* automatycznie dodaje znak powrotu karetki (`\r`) do wysyłanego ciągu. Prosty skrypt mógłby wyglądać tak:

```
ogin: vstout ssword: catch22
```

Prawdopodobnie zauważyłeś, że pola oczekiwanego ciągu nie zawierają całych monitów. Dzięki temu logowanie się powiedzie, nawet jeżeli zdalny host prześle *Login:* zamiast *login:*. Jeżeli oczekiwany lub wysyłany ciąg zawiera spacje albo inne białe znaki, musisz wziąć go w cudzysłów.

uucico pozwala także na swego rodzaju wykonywanie warunkowe. Powiedzmy, że *getty* na zdalnej maszynie musi być wyzerowany przed wysłaniem monitu. W tym celu możesz dołączyć do oczekiwanego ciągu poddialog, wywoływany znakiem -. Poddialog jest wykonywany wtedy, gdy główny oczekiwany ciąg nie zostanie dopasowany, tj. zostanie przekroczony czas oczekiwania. Jednym ze sposobów na użycie tej funkcji jest wysłanie sygnału BREAK, jeżeli zdalny ośrodek nie wyświetli monitu logowania. Poniższy przykład pokazuje skrypt dialogowy ogólnego przeznaczenia, który powinien działać także w przypadku, gdy musisz nacisnąć [Enter] przed pojawieniem się monitu. Pusty pierwszy argument ("") mówi UUCP, by na nic nie czekało, ale działało dalej, wysyłając kolejny ciąg znaków:

```
"" \n\r\d\r\n\c ogin:-BREAK-ogin: vstout ssword: catch22
```

W skrypcie dialogowym może wystąpić kilka znaków unikowych (ang. *escape*) i specjalnych ciągów. Oto częściowa lista znaków dopuszczalnych w oczekiwanym ciągu:

`""`
: Ciąg pusty. Mówi *uucico*, by nie czekało na nic, ale natychmiast wysłało następny ciąg.

`\t`
: Znak tabulacji.

`\r`
: Znak powrotu karetki.

`\s`
: Spacja. Potrzebne do umieszczania spacji w ciągu dialogowym.

`\n`
: Znak nowego wiersza.

`\\`
: Odwrotny ukośnik.

Poza powyższymi znakami w wysyłanych ciągach znaków dopuszczalne są poniższe znaki unikowe:

`EOT`
: Znak końca transmisji (^*D*).

`BREAK`
: Znak przerwania.

`\c`
: Zapobiega wysłaniu znaku powrotu karetki na końcu ciągu.

`\d`
: Opóźnienie wysyłania o 1 sekundę.

`\E`
: Włączenie sprawdzania echa. Nakazuje *uucico* czekać na echo wszystkiego co wyśle, zanim będzie prowadzić dalszy dialog. Przydaje się w dialogach modemowych (co zobaczymy później). Sprawdzanie echa jest domyślnie wyłączone.

`\e`
: Zablokowanie sprawdzania echa.

`\K`
: To samo co `BREAK`.

`\p`
: Czekanie przez ułamek sekundy.

Alternatywy

Czasem chcesz mieć kilka wpisów dla jednego systemu, na przykład jeżeli można do niego dotrzeć przez różne linie modemowe. W przypadku UUCP Taylora możesz to zrobić, definiując tak zwane *alternatywy* (ang. *alternates*).

Wpis alternatywny zachowuje wszystkie ustawienia głównego wpisu charakteryzujące go system i określa tylko te wartości, które powinny być zmienione lub dodane. Wpis alternatywny jest umieszczany za wpisem opisującym system, po wierszu ze słowem `alternate`.

Aby używać dwóch numerów telefonu do systemu **pablo**, powinieneś zmodyfikować jego opis w pliku *sys* do następującej postaci:

```
system   pablo
phone   123-456
.. wpisy podobne do powyższych ...
alternate
phone   123-455
```

Dzwoniąc do **pablo**, *uucico* najpierw używa numeru 123-456, a jeżeli się nie dodzwoni, próbuje numeru alternatywnego.

Wyznaczanie czasów dzwonienia

Taylor UUCP posiada szereg sposobów wyznaczenia godzin, o których są realizowane połączenia z systemem zdalnym. Możesz ich potrzebować ze względu na ograniczenia stawiane przez system zdalny w godzinach roboczych lub po prostu

by uniknąć godzin o wysokich cenach rozmów. Zauważ, że zawsze możliwe jest ominięcie ograniczeń czasowych przez podanie *uucico* opcji *–S* lub *–f*.

Domyślnie Taylor UUCP nie pozwala na połączenia o dowolnych godzinach, a więc *musisz* w pliku *sys* wymienić jakieś godziny. Jeżeli nie dbasz o ograniczenia czasowe, możesz użyć w swoim pliku *sys* opcji *time* z wartością `Any`.

Najprostszym sposobem na ograniczenie godzin dzwonienia jest dołączenie wpisu `time`, a za nim ciągu składające go się z pól opisujących dzień i godzinę. Dzień może być kombinacją `Mo`, `Tu`, `We`, `Th`, `Fr`, `Sa` i `Su`. Możesz także użyć `Any`, `Never` lub `Wk` dla dni roboczych. Czas składa się z dwóch wartości w postaci 24-godzinnej, oddzielonych myślnikiem. Określają one okres, w którym mogą być wykonywane połączenia. Połączenie tych leksemów jest zapisywane bez spacji pomiędzy nimi. Dowolne określenia dnia i godziny mogą być pogrupowane razem i oddzielone przecinkami w następujący sposób:

```
time          MoWe0300-0730,Fr1805-2200
```

Ten przykład pokazuje, że połączenia mogą być realizowane w poniedziałki i środy od 3:00 do 7:30 oraz w piątki od 18:05 do 22:00. Gdy pole opisujące czas obejmuje północ, powiedzmy `Mo1830-0600`, w rzeczywistości oznacza to poniedziałek pomiędzy północą a szóstą rano oraz pomiędzy 18:30 i północą.

Specjalne ciągi opisujące czas, `Never` i `Any`, oznaczają odpowiednio, że połączenia nie mogą być realizowane lub mogą być realizowane o dowolnej godzinie.

UUCP Taylora posiada również szereg specjalnych leksemów, których możesz używać w opisie czasu, jak `NonPeak` i `Night`. Te szczególne leksemy to odpowiednio skróty od `Any2300-0800,SaSu0800-1700` i `Any1800-0700,SaSu`.

Polecenie *time* przyjmuje opcjonalnie drugi argument opisujący w minutach czas powtarzania. Gdy próba nawiązania połączenia się nie powiedzie, *uucico* poczeka z wykonaniem kolejnej próby przez pewien okres czasu. Na przykład gdy ustawisz czas powtarzania na 5 minut, *uucico* będzie odmawiać dzwonienia do zdalnego systemu przez 5 minut, poczynając od ostatniej nieudanej próby. Domyślnie *uucico* używa schematu wykładniczego, gdzie okres przed ponowną próbą zwiększa się przy każdym kolejnym niepowodzeniu.

Polecenie *timegrade* pozwala na powiązanie czasów z maksymalnym stopniem buforowania. Na przykład załóżmy, że we wpisie `system` masz następujące polecenia *timegrade*:

```
timegrade       N Wk1900-0700,SaSu
timegrade       C Any
```

Taki zapis oznacza, że zadania o stopniu buforowania C lub wyższym (zwykle poczta jest kolejkowana ze stopniem B lub C) zostaną przesłane po zestawieniu połączenia, natomiast grupy dyskusyjne (zwykle zakolejkowane ze stopniem N) są przesyłane tylko w nocy i w weekendy.

Podobnie jak *time*, tak i *timegrade* posiada trzeci argument opisujący przerwę (w minutach) przed ponowieniem próby.

Jednak są tu pewne zastrzeżenia co do stopnibu forowania. Przede wszystkim opcja *timegrade* dotyczy tylko tego, co wysyła *twój* system. System zdalny może wciąż przesyłać, co chce. Możesz użyć opcji *call-timegrade*, aby jawnie zażądać wysyłania jedynie zadań o stopniu wyższym niż zadany, ale nie ma gwarancji, że żądanie to zostanie wysłuchane*.

Podobnie pole `timegrade` nie jest sprawdzane, gdy dzwoni system zdalny, a więc wszelkie zadania zakolejkowane dla niego zostaną wysłane. Jednak system zdalny może w jawny sposób zażądać, aby twoje *uucico* ograniczyło się do pewnego stopnia buforowania.

Identyfikowanie dostępnych urządzeń poprzez plik port

Plik *port* informuje *uucico* o dostępnych portach. Są to zwykle porty modemowe, ale obsługiwane są także inne typy portów, na przykład bezpośrednie łącza szeregowe i gniazda TCP.

Podobnie jak plik *sys*, tak i *port* składa się z oddzielnych wpisów rozpoczynających się od słowa kluczowego `port`, za którym następuje nazwa portu. Nazwa nie musi być unikalna. Jeżeli istnieje kilka portów o tej samej nazwie, *uucico* będzie je sprawdzać po kolei, aż znajdzie ten, który nie jest właśnie używany.

Zaraz za poleceniem *port* powinna występować dyrektywa `type` wskazująca typ opisywanego portu. Dopuszczalne typy to `modem`, `direct` dla łączy bezpośrednich i `tcp` dla gniazd TCP. Jeżeli brakuje polecenia *port*, domyślnym typem jest modem.

Tutaj omówimy tylko porty modemowe. Porty TCP i łącza bezpośrednie są opisane dalej.

W przypadku portów modemowych i bezpośrednich, w dyrektywie `device` musisz podać urządzenie, przez które chcesz dzwonić. Zwykle jest to nazwa specjalnego pliku urządzenia w katalogu */dev*, na przykład */dev/ttyS1*.

W przypadku modemu wpis *port* określa również, jakiego typu modem jest podłączony. Różne typy modemów muszą być odpowiednio skonfigurowane. Nawet modemy, które deklarują kompatybilność ze standardem Hayesa, nie zawsze są naprawdę kompatybilne ze sobą. Dlatego musisz poinformować *uucico*, jak zainicjować modem i zadzwonić na żądany numer. UUCP Taylora przechowuje opis wszystkich urządzeń w pliku *dial*. Aby użyć któregoś z nich, musisz podać jego nazwę za pomocą polecenia *dialer*.

Czasem będziesz chciał używać modemu na różne sposoby, w zależności od tego, do jakiego systemu dzwonisz. Na przykład niektóre starsze modemy tracą orientację, gdy szybkie modemy próbują się łączyć z prędkością 56 kilobitów na sekundę. Po prostu zrywają połączenie, zamiast negocjować na przykład prędkość 9600 bitów na sekundę. Gdy wiesz, że ośrodek **drop** używa takich mało inteligentnych modemów, musisz inaczej skonfigurować swój modem, gdy tam dzwonisz. Potrzebujesz dodatkowego wpisu w pliku *port*, który wskazuje inny typ urządzenia dzwoniące-

* Jeżeli w systemie zdalnym działa UUCP Taylora, to żądanie zostanie wysłuchane.

go. W tym przypadku możesz nadać nowemu portowi inną nazwę, jak `serial1-slow` i użyć dyrektywy `port` we wpisie dla systemu **drop** w pliku *sys*.

Portyna jle piej rozróżnia się na podstawie obsługiwanych przez nie prędkości. Na przykład dwa opisy portów dla powyższej sytuacji mogłyby wyglądać tak:

```
# NakWell modem; po□□czenie przy du□ej pr□dko□ci
port      serial1        # nazwa portu
type      modem          # port modemu
device    /dev/ttyS1     # to jest COM2
speed     115200         # obs□ugiwana pr□dko□□
dialer    nakwell        # normalny typ
# NakWell modem; po□□czenie przy niskiej pr□dko□ci
port      serial1        # nazwa portu
type      modem          # port modemu
device    /dev/ttyS1     # to jest COM2
speed     9600           # obs□ugiwana pr□dko□□
dialer    nakwell-slow   # nie próbuj szybkiego po□□czenia
```

W opisie systemu **drop** jako nazwa portu widnieje teraz `serial1`, ale będzie obsługiwane tylko żądania połączenia z prędkością 9600 bitów na sekundę. *uucico* automatycznie użyje drugiego portu. Wszystkie pozostałe ośrodki, które łączą się z prędkością 115200 bitów na sekundę, będą używały pierwszego wpisu. Domyślnie będzie używany pierwszy wpis, który obsługuje odpowiednią prędkość.

Jak dzwonić pod zadany numer używając pliku dial

Plik *dial* opisuje, w jaki sposób są używane różne typy urządzeń. Tradycyjnie UUCP rozmawia z urządzeniami, a nie z modemami, ponieważ dawniej zwykle dostępne było jedno (drogie!) urządzenie dzwoniące, które obsługiwało cały zestaw modemów. Obecnie większość modemów ma wbudowaną obsługę dzwonienia, a więc to rozróżnienie przestaje być ważne.

Niezależnie od tego, czy mamy do czynienia z urządzeniami dzwoniącymi, czy z modemami, różne ich typy mogą wymagać odmiennej konfiguracji. Każdy z nich możesz opisać w pliku *dial*. Wpisy w *dial* rozpoczynają się od polecenia *dialer* zawierającego nazwę urządzenia.

Najważniejszy wpis po *dialer* to dialog modemu opisywany przez polecenie *chat*. Podobnie do dialogu logowania, składa się z wielu ciągów znaków, które *uucico* wysyła do urządzenia dzwoniącego, oraz z odpowiedzi, których oczekuje. Zwykle jest on używany do ustawienia modemu w jakimś znanym stanie i wykręcenia numeru. Poniższy przykładowy wpis `dialer` pokazuje typowy dialog modemu dla modemów kompatybilnych ze standardem Hayesa:

```
# NakWell modem; po□□czenie przy du□ej pr□dko□ci
dialer         nakwell      # nazwa urz□dzenia
chat           "" AT&F OK\r ATH1E0Q0 OK\r ATDT\T CONNECT
chat-fail      BUSY
chat-fail      ERROR
chat-fail      NO\sCARRIER
dtr-toggle     true
```

Dialog modemu rozpoczyna się od " ", czyli oczekiwanego ciągu pustego. *uucico* w tej sytuacji wysyła natychmiast pierwsze polecenie *AT&F*. Jest to polecenie Hayesa usta-

wiające modem w konfiguracji fabrycznej. *uucico* następnie czeka, aż modem wyśle `OK`, i przekazuje następne polecenie, które z kolei wyłącza lokalne echo i tym podobne. Po odebraniu z modemu kolejnego `OK`, *uucico* wysyła polecenie dialogu *ATDT*. Sekwencja unikowa `\T` w ciągu znaków zostaje zastąpiona numerem telefonu odczytanym z wpisu w pliku *sys*. *uucico* następnie czeka, aż modem zwróci ciąg `CONNECT`, który sygnalizuje, że połączenie z modemem zdalnym zostało pomyślnie nawiązane.

Czasem modemom nie udaje się połączyć z systemem zdalnym. Na przykład jeżeli drugi system akurat z kimś się łączy i linia jest zajęta. W takiej sytuacji modem zwraca błąd wskazujący powód niepowodzenia. Dialogi modemowe nie są w stanie obsłużyć takich błędów. *uucico* dalej czeka na oczekiwany ciąg znaków, aż upłynie zadany czas. W pliku log UUCP w takiej sytuacji pokazany jest jedynie komunikat „timed out in chat script" (upłynął czas oczekiwania w skrypcie dialogowym), zamiast wskazania konkretnego powodu.

Jednak UUCP Taylora pozwala na poinformowanie *uucico* o tych komunikatach błędów. Służy do tego polecenie *chat-fail* pokazane powyżej. Gdy *uucico* wykryje ciąg nieudanego dialogu podczas dialogu modemowego, przerywa połączenie i loguje komunikat do pliku log UUCP.

Ostatnie polecenie w pokazanym powyżej przykładzie informuje UUCP, by przed rozpoczęciem dialogu modemowego przełączyło linię sterującą DTR (*Data Terminal Ready*). Normalnie urządzenie szeregowe uaktywnia DTR, gdy proces otworzy urządzenie, aby poinformować podłączony modem, że ktoś chce z nim się połączyć. Funkcja `dtr-toggle` deaktywuje DTR, czeka chwilę i aktywuje go ponownie. Wiele modemów można skonfigurować tak, by reagowały na deaktywację DTR przez rozłączenie, przejście do trybu wprowadzania poleceń czy wyzerowanie się*.

Przesyłanie UUCP przez TCP

Użycie UUCP do przesyłania danych przez TCP może brzmieć absurdalnie, ale nie jest to zły pomysł, szczególnie gdy przesyłasz duże ilości danych, jak grupy dyskusyjne Usenetu. Na łączach opartych o TCP grupy są przeważnie wymieniane przez protokół NNTP, w którym żądane artykuły są przesyłane pojedynczo, bez kompresji lub innej optymalizacji. Choć technika ta sprawdza się dla dużych ośrodków o kilku równoległych połączeniach grup, to nie jest zbyt lubiana przez małe ośrodki, które odbierają swoje grupy przez relatywnie wolne połączenia, jak ISDN. Te ośrodki zwykle będą chciały połączyć jakość TCP z zaletami wysyłania grup w dużych porcjach, które mogą być kompresowane i przesyłane bez nadmiarowych informacji. Zwykle przesyła się je za pomocą UUCP przez TCP.

W pliku *sys* określasz system wywoływany przez TCP w następujący sposób:

```
system      gmu
address     news.groucho.edu
time        Any
port        tcp-conn
chat        ogin: vstout word: clouseau
```

* Niektóre modemy tego nie lubią i czasami się zawieszają.

Polecenie *address* zawiera adres IP hosta lub jego pełną nazwę domenową. Odpowiedni wpis *port* wyglądałby tak:

```
port        tcp-conn
type        tcp
service     540
```

Wpis ten mówi, że połączenie TCP powinno być używane, gdy wpis *sys* zawiera ciąg `tcp-conn` i że *uucico* powinno próbować łączyć się z portem 540 hosta zdalnego w sieci TCP. Jest to domyślny port usługi UUCP. Zamiast numeru portu, w poleceniu *service* możesz podać także jego symboliczną nazwę. Odpowiadający jej numer portu będzie poszukiwany w pliku */etc/services*. Powszechnie używaną nazwą dla usług UUCP jest `uucpd`.

Używanie połączenia bezpośredniego

Załóżmy, że używasz bezpośredniego łącza przy komunikacji systemu **vstout** z systemem **tiny**. Podobnie jak w przypadku modemu, musisz stworzyć w pliku *sys* odpowiedni wpis opisujący system. Polecenie *port* identyfikuje port, do którego jest podłączony system `tiny`:

```
system      tiny
time        Any
port        direct1
speed       38400
chat        ogin: cathcart word: catch22
```

W pliku *port* musisz opisać port szeregowy dla połączenia bezpośredniego. Wpis `dialer` jest zbyteczny, ponieważ nie ma potrzeby dzwonienia:

```
port        directl
type        direct
speed       38400
device      /dev/ttyS1
```

Kontrola dostępu do funkcji UUCP

UUCP jest systemem dość elastycznym. Dlatego trzeba kontrolować uważnie dostęp do jego funkcji, aby zapobiec celowym lub przypadkowym nadużyciom. Podstawowe funkcje UUCP, którymi powinien się zająć administrator, to wykonywanie zdalnych poleceń, przesyłanie plików i przekazywanie. UUCP Taylora pozwala na pewne ograniczenie działania każdej z funkcji, którym musi się podporządkować zdalny host UUCP. Starannie dobrawszy prawa dostępu, administrator UUCP może być pewny, że host jest bezpieczny.

Wykonywanie polecenia

Zadaniem UUCP jest przypilnowanie kopiowania plików z jednego systemu do innego i żądanie wykonania pewnych poleceń na hostach zdalnych. Oczywiście ty jako administrtor chciałbyś kontrolować prawa przydzielane innym systemom, ponieważ pozwalanie im na wykonanie w twoim systemie dowolnego polecenia zdecydowanie nie jest dobrym pomysłem.

Domyślnie jedynymi poleceniami, jakie UUCP Taylora pozwala wykonywać innym systemom na twoim komputerze, są *rmail* i *rnews*, powszechnie używane do wymiany poczty i grup dyskusyjnych Usenetu przez UUCP. Aby zmienić zestaw poleceń dla jakiegoś systemu, możesz użyć słowa kluczowego `commands` w pliku *sys*. Możesz też ograniczyć ścieżkę poszukiwań jedynie do katalogów zawierających dozwolone polecenia, używając dyrektywy `command-path`. Na przykład możesz pozwolić systemowi **pablo** na wykonywanie, oprócz poleceń *rmail* i *rnews*, także polecenia *bsmtp**:

```
system      pablo
...
commands    rmail rnews bsmtp
```

Przesyłanie plików

UUCP Taylora pozwala również dostosować przesyłanie plików dokładnie do twoich potrzeb. Możesz wyłączyć przesyłanie plików do i z jakiegoś systemu. Po prostu ustaw `request` na wartość `no`, a system zdalny nie będzie w stanie ani odbierać, ani wysyłać żadnych plików z twojego systemu. Podobnie możesz zakazać użytkownikom przesyłania plików do i z systemu, ustawiając opcję `transfer` na `no`. Domyślnie użytkownicy obu systemów, lokalnego i zdalnego, nie będą mogli przesyłać plików w żadną stronę.

Poza tym możesz skonfigurować katalogi, do których i z których mogą być kopiowane pliki. Zwykle ograniczasz dostęp zdalnych systemów tylko do jednego drzewa katalogów, ale wciąż pozwalasz użytkownikom na wysyłanie plików z ich katalogu macierzystego. Przeważnie zdalni użytkownicy mają prawo pobierać pliki jedynie z publicznego katalogu UUCP, */var/spool/uucppublic*. Jest to tradycyjne miejsce publicznego udostępniania plików, podobnie jak serwery FTP w Internecie**.

UUCP Taylora udostępnia cztery różne polecenia do konfigurowania katalogów do wysyłania i odbierania plików. Są to: *local-send* – określa listę katalogów, z których użytkownik może wysyłać pliki przez UUCP, *local-receive* – określa listę katalogów, z których użytkownik może pobierać pliki, *remote-send* oraz *remote-receive*, które działają analogicznie, ale dla żądań z systemów zdalnych. Przyjrzyj się poniższemu przykładowi:

```
system            pablo
...
local-send        /home ~
local-receive     /home ~/receive
remote-send      ~ !~/incoming !~/receive
remote-receive   ~/incoming
```

Polecenie *local-send* pozwala użytkownikom twojego hosta na wysyłanie dowolnych plików z katalogu poniżej */home* i z publicznego katalogu UUCP do systemu **pablo**. Polecenie *local-receive* pozwala na umieszczanie plików w dostępnym do publiczne-

* *bsmtp* jest używane do dostarczania poczty w systemie wsadowym SMTP.

** Możesz użyć znaku tyldy (~), by odwołać się do katalogu publicznego UUCP, ale tylko w plikach konfiguracyjnych UUCP. Poza nimi tylda zwykle oznacza katalog macierzysty użytkownika

go zapisu katalogu *receive* w *uucppublic* lub w dostępnych do publicznego zapisu katalogach poniżej */home*. Dyrektywa *remote-send* pozwala systemowi **pablo** na żądanie plików z katalogu */var/spool/uucppublic*, poza plikami zawartymi w katalogach *incoming* i *receive*. Jest to sygnalizowane *uucico* przez poprzedzenie nazw katalogów wykrzyknikami. Ostatni wiersz pozwala **pablo** na umieszczanie plików w katalogu **incoming**.

Głównym problemem przy przesyłaniu plików za pomocą UUCP jest to, że pliki są odbierane tylko do katalogów, które są publicznie dostępne do zapisu. Może to skusić niektórych użytkowników do zastawiania pułapek na innych. Jednak nie da się rozwiązać tego problemu inaczej, niż przez całkowite zablokowanie przesyłania plików do UUCP.

Przekazywanie

UUCP oferuje mechanizm pozwalający na inicjację przesyłania plików w twoim imieniu. Załóżmy na przykład, że twój system ma dostęp *uucp* do systemu **seci**, ale nie ma dostępu do systemu **uchile**. Za pośrednictwem wspomnianego mechanizmu możesz poprosić **seci**, aby odebrał dla ciebie pliki z **uchile** i wysłał je do twojego systemu. Można to zrobić poniższym poleceniem:

```
$ uucp -r seci!uchile!~/find-ls.gz ~/uchile.files.gz
```

Ta technika przekazywania zadań przez kilka systemów jest nazywana po prostu *przekazywaniem* (ang. *forwarding*). W twoim systemie UUCP możesz ograniczyć usługę przekazywania do kilku hostów, o których wiesz, że nie nabiją ci ogromnego rachunku telefonicznego przy ściąganiu najnowszych plików źródłowych X11R6.

Domyślnie UUCP Taylora w ogóle wyłącza przekazywanie. Aby je włączyć dla jakiegoś systemu, możesz użyć polecenia *forward*. Polecenie to wypisuje listę ośrodków, które mogą żądać od ciebie przekazania zadań do nich i od nich. Na przykład administrator UUCP systemu **seci** mógłby dodać następujące wiersze do pliku *sys*, by pozwolić systemowi **pablo** na ściąganie plików z **uchile**:

```
########################
# pablo
system          pablo
...
forward          uchile
########################
# uchile
system           uchile
...
forward-to       pablo
```

Wpis `forward-to` dla **uchile** jest potrzebny po to, by wszystkie odbierane pliki były przekazywane do **pablo**. W przeciwnym razie UUCP by je odrzucało. Wpis ten wykorzystuje odmianę polecenia *forward* pozwalającą **uchile** na wysyłanie plików tylko do **pablo** przez **seci**, a nie w inny sposób.

Aby zezwolić na przekazywanie do dowolnego systemu, użyj specjalnego słowa kluczowego `ANY` (wymagane duże litery).

Konfigurowanie systemu do przyjmowania połączeń komutowanych

Jeśli chcesz skonfigurować swój system tak, aby przyjmował połączenia komutowane, musisz zezwolić na logowanie do twojego portu szeregowego i dostosować niektóre pliki systemowe do obsługi kont UUCP. Omówimy to w tym podrozdziale.

Zapewnienie kont UUCP

Na początek musisz skonfigurować konta użytkownika, które pozwolą zdalnym ośrodkom logować się do twojego systemu i realizować połączenie UUCP. Musisz stworzyć oddzielną nazwę dla każdego systemu, który będzie się z tobą łączył. Przy konfigurowaniu konta dla systemu **pablo** możesz wykorzystać nazwę użytkownika **Upablo**. Nie ma tu tajżadnych ustalonych zasad tworzenia nazw użytkowników, a więc mogą być dowolne; wygodniej jest, jeżeli nazwa użytkownika wiąże się w jakiś sposób z nazwą hosta zdalnego.

W przypadku systemów, które wdzwaniają się przez port szeregowy, zwykle dodajesz konta w pliku */etc/passwd*. Dobrze jest umieścić wszystkie identyfikatory UUCP w specjalnej grupie, na przykład **uuguest**. Katalog macierzysty kont powinien być ustawiony na publiczny katalog bufora */var/spool/uucppublic*. Powłoka logowania musi być ustawiona na *uucico*.

Aby obsłużyć systemy UUCP łączące się do ciebie przez TCP, musisz skonfigurować *inetd* tak, aby obsługiwał połączenia przychodzące na port `uucp`. W tym celu dodaje się poniższy wiersz do pliku */etc/inetd.conf**:

```
uucp   stream   tcp   nowait   root   /usr/sbin/tcpd   /usr/lib/uucp/uucico -l
```

Opcja *–l* powoduje, że *uucico* przeprowadza własną procedurę uwierzytelniania. Pyta o nazwę użytkownika i hasło, tak jak standardowy program *login*, ale wykorzystuje swoją prywatną bazę haseł, zamiast */etc/passwd*. Prywatny plik haseł nazywa się */etc/uucp/passwd* i zawiera połączone w pary: nazwy użytkowników i hasła:

```
Upablo   IslaNegra
Ulorca   co'rdoba
```

Plik ten musi być własnością **uucp** i mieć prawa dostępu 600.

Czy ta baza danych wygląda na tyle sensownie, byś chciał jej używać także do zwykłego logowania przez łącza szeregowe? Oczywiście, w pewnych sytuacjach możesz. Potrzebujesz jedynie programu *getty*, który w przypadku użytkowników UUCP ma możliwość wywołania *uucico* zamiast */bin/login***. Wywołanie *uucico* wygląda tak:

```
/usr/lib/uucp/uucico -l -u użytkownik
```

* Zauważ, że *tcpd* zwykle ma tryb 700, a więc musisz go wywoływać jako użytkownik **root**, a nie **uucp**. *tcpd* jest omówione dokładniej w rozdziale 12, *Ważne funkcje sieciowe*.

** Dobrze nadaje się do tego *mgetty* Gerta Doeringa. Działa na różnych platformach, włącznie z SCO Unix, AIX, SunOS, HP-UX i Linuksem.

Opcja *–u* mówi, by *uucico* używał zadanej nazwy użytkownika, zamiast o nią pytać*.

Aby zabezpieczyć twoich użytkowników UUCP przed dzwoniącymi, którzy mogliby podać fałszywą nazwę systemu i przejąć całą pocztę, powinieneś dodać polecenia *called-login* do każdego wpisu systemu w pliku *sys*. Wyjaśniamy to w następnym podrozdziale.

Zabezpieczanie się przed kanciarzami

Głównym problemem UUCP jest to, że dzwoniący system może podać fałszywą nazwę. Po zalogowaniu się system podaje nazwę, ale serwer nie ma sposobu na jej sprawdzenie. Dlatego atakujący mógłby zalogować się na swoje konto UUCP, udawać kogoś innego i pobrać pocztę przeznaczoną dla innego ośrodka. Jest to szczególnie problematyczne, gdy oferujesz logowanie anonimowe, gdzie hasło jest publicznie dostępne.

Musisz bronić się przed oszustami. Każdy system powinien używać jakiejś nazwy użytkownika podanej w `called-login` w pliku *sys*. Przykładowy wpis mógłby wyglądać tak:

```
system      pablo
... typowe opcje...
called-login   Upablo
```

Rezultat jest taki, że gdy system się zaloguje i twierdzi, że nazywa się **pablo**, *uucico* sprawdza, czy zalogował się jako **Upablo**. Jeżeli nie, system dzwoniący jest wyłączany, a połączenie – zrywane. Dopisywanie polecenia *called-login* do każdego wpisu systemowego w twoim pliku *sys* powinno wejść ci w krew. Ważne jest, byś zrobił to dla *wszystkich* systemów z twojego pliku *sys*, bez względu na to, czy kiedykolwiek będą dzwoniły do twojego ośrodka, czy nie. Dla tych, które nigdy nie dzwonią, powinieneś ustawić `called-login` na jakąś całkowicie fałszywą nazwę użytkownika, jak **neverlogsin**.

Bądź paranoikiem: sprawdzanie licznika połączeń

Innym sposobem ochrony swojego systemu i wykrywania oszustów jest użycie *licznika wywołań*. Pomaga on zabezpieczyć się przed intruzami, którzy w jakiś sposób zdobyli hasło i mogą się zalogować do twojego systemu UUCP.

Sprawdzanie licznika połączeń polega na tym, że obie maszyny śledzą liczbę zrealizowanych do tej pory połączeń. Licznik jest zwiększany przy każdym połączeniu. Po zalogowaniu się dzwoniący wysyła swój kolejny numer, a odbiorca porównuje go z własnym numerem. Jeżeli się nie zgadzają, próba połączenia kończy się odmową. Jeżeli pierwsza liczba zostanie wybrana losowo, intruz będzie miał problem, by poprawnie zgadnąć kolejny numer połączenia.

Jednak sprawdzanie licznika połączeń to coś więcej. Nawet jeżeli jakiś mądrala wykryłby twój numer połączenia i twoje hasło, dowiedziałbyś się o tym. Gdy atakujący dzwoni do twojego węzła pocztowego UUCP i kradnie pocztę, numer porządkowy

* Tej opcji nie ma w wersji 1.04.

w węźle zwiększa się o jeden. Następnie, gdy *ty* zadzwonisz to twojego węzła i spróbujesz się zalogować, zdalne *uucico* odmówi ci, po nieważ numery nie będą się zgadzały!

Jeżeli włączyłeś sprawdzanie licznika połączeń, powinieneś przeglądać regularnie pliki log, poszukując komunikatów błędów, które informują o potencjalnych atakach. Jeżeli twój system od rzuca numer połączenia odebrany z systemu dzwoniącego, *uucico* umieszcza w pliku log komunikat o treści „Out of sequence call rejected" (odrzucono połączenie o złym numerze porządkowym). Jeżeli twój system zostanie odrzucony przez węzeł ze względu na zły numer, w pliku log pojawi się komunikat „Handshake failed (RBADSEQ)" (Uzgadnianie się nie powiodło).

Aby włączyć sprawdzanie licznika połączeń, dodaj poniższe polecenie do opisu systemu:

```
# w□□czenie sprawdzania licznika po□□cze□
sequence      true
```

Ponadto musisz stworzyć plik zawierający sam numer połączenia. UUCP Taylora przechowuje ten numer w pliku *.Sequence* w katalogu buforowym hosta zdalnego. Plik *musi* być własnością użytkownika **uucp** i musi mieć prawa 600 (to znaczy prawa czytania i zapisu tylko dla użytkownika **uucp**). Najlepiej jest zainicjować ten plik wcześniej uzgodnioną wartością losową. Prosty sposób na utworzenie tego pliku jest następujący:

```
# cd /var/spool/uucp/pablo
# echo 94316 > .Sequence
# chmod 600 .Sequence
# chown uucp.uucp .Sequence
```

Oczywiście zdalny ośrodek musi także włączyć sprawdzanie licznika połączeń i rozpocząć od tego samego numeru co ty.

Anonimowe UUCP

Gdybyś chciał dać anonimowy dostęp UUCP do swojego systemu, musiałbyś najpierw stworzyć specjalne konto zgodnie z tym, co wspomnieliśmy wcześniej. Powszechnie tworzy się konto o nazwie i haśle **uucp**.

Ponadto musisz skonfigurować kilka opcji bezpieczeństwa dla nieznanych systemów. Na przykład możesz zakazać im wykonywania pewnych poleceń w twoim systemie. Jednak nie możesz ustawić tych parametrów w pliku sys, ponieważ polecenie *system* wymaga podania nazwy systemu, a tej nie znasz. UUCP Taylora rozwiązuje ten problem przez polecenie *unknown*. Może być ono używane w pliku *config* do określenia dowolnego polecenia, które zwykle pojawia się w opisie systemu:

```
unknown       remote-receive ~/incoming
unknown       remote-send ~/pub
unknown       max-remote-debug none
unknown       command-path /usr/lib/uucp/anon-bin
unknown       commands rmail
```

Utrudnia to nieznanym systemom pobieranie plików z katalogu *pub* i wrzucanie plików do katalogu *incoming* w */var/spool/uucppublic*. Następny wiersz powoduje, że *uucico* będzie ignorować wszelkie żądania od systemów odnoszące się do lokalnego

włączenia debugowania. Ostatnie dwa wiersze zezwalają nieznanym systemom na wywołanie *rmail*. Jednak podanie ścieżka ze zwala *uucico* szukać polecenia *rmail* tylko w katalogu prywatnym *anon-bin*. To ograniczenie pozwala na udostępnienie specjalnego polecenia *rmail*, które na przykład przekazuje całą pocztę superużytkownikowi do sprawdzenia. Pozwala to anonimowym użytkownikom na dotarcie do właściciela systemu, ale jednocześnie zapobiega wysyłaniu poczty do innych ośrodków.

Aby włączyć anonimowe UUCP, musisz w pliku *config* podać przynajmniej jedną dyrektywę `unknown`. W przeciwnym razie *uucico* odrzuci wszystkie nieznane systemy.

Protokoły niskiego poziomu w UUCP

Aby wynegocjować z drugą stroną sterowanie sesją i przesyłanie plików, *uucico* używa zestawu standardowych komunikatów. Często nazywa się to *protokołem wysokiego poziomu*. W czasie fazy inicjacyjnej i fazy zawieszania są one wysyłane w postaci prostych ciągów znaków. Jednak w czasie fazy rzeczywistego przesyłania używany jest dodatkowo protokół niskiego poziomu, przeważnie przezroczysty dla wyższych poziomów. Protokół ten daje pewne dodatkowe możliwości, jak sprawdzanie błędów w wysyłanych danych w przypadku zawodnych łączy.

Przegląd protokołów

UUCP jest używane z różnymi typami połączeń, jak łącza szeregowe, TCP lub czasem nawet X.25. Korzystne jest przesyłanie UUCP w protokołach stworzonych specjalnie dla niższych protokołów sieciowych. Ponadto kilka implementacji UUCP zawiera różne protokoły, które robią z grubsza to samo.

Protokoły można podzielić na dwie kategorie: *strumieniowe* i *pakietowe*. Protokoły strumieniowe przesyłają plik w całości, obliczając ewentualnie jego sumę kontrolną. W takiej sytuacji prawie nie istnieje nadmiarowość, ale wymagane jest niezawodne połączenie, ponieważ najmniejszy błąd powoduje, że cały plik musi być przesłany ponownie. Te protokoły są przeważnie używane w połączeniach TCP, ale nie nadają się do użytku w liniach telefonicznych. Choć nowoczesne modemy do skonale korygują błędy, nie są idealne. Nie ma też możliwości wykrycia błędów występujących pomiędzy twoim komputerem a modemem.

Natomiast protokoły pakietowe dzielą plik na kilka równych części. Każdy pakiet jest wysyłany i odbierany niezależnie, obliczana jest suma kontrolna, a do nadawcy zwracane jest potwierdzenie. Aby usprawnić działanie tych protokołów, opracowano protokoły przesuwnego okna (ang. *sliding-window protocols*), których zadaniem jest przyjęcie pewnej ograniczonej liczby (okno) zaległych potwierdzeń w dowolnej chwili. Znacznie skraca to czas przestoju *uucico* w czasie przesyłania. Relatywnie duża nadmiarowość w porównaniu z protokołami strumieniowymi powoduje, że protokoły pakietowe są nieefektywne we współpracy z TCP, natomiast są idealne dla linii telefonicznych.

Rozszerzenie ścieżki danych także nie jest bez znaczenia. Czasem niemożliwe jest wysyłanie znaków 8-bitowych przez łącze szeregowe. Na przykład połączenie mogłoby prowadzić do uproszczonego terminalaserwera, który obcina ostatni bit. Gdy przesyłasz 8-bitowe znaki przez 7-bitowe połączenie, muszą być one maskowane. W najgorszym przypadku cytowanie dwukrotnie zwiększa liczbę przesyłanych danych, choć można to wyrównać kompresją realizowaną sprzętowo. Linie, które mogą przesyłać dowolne znaki 8-bitowe, są zwykle nazywane *8-bit clean*. Takie właśnie są wszystkie połączenia TCP oraz większość połączeń modemowych.

UUCP Taylora w wersji 1.06 obsługuje szereg protokołów UUCP. Najważniejsze z nich to:

g
: Jest to najpopularniejszy protokół i powinien być rozumiany przez praktycznie wszystkie *uucico*. Gruntownie sprawdza błędy i doskonale nadaje się do szumiących linii telefonicznych. *g* wymaga połączenia 8-bitowego. Jest protokołem pakietowym, wykorzystującym technikę przesuwnego okna.

i
: Jest to pakietowy protokół dwukierunkowy, który może wysyłać i odbierać pliki w tym samym czasie. Wymaga połączenia w pełnym dupleksie i 8-bitowej ścieżki danych. Aktualnie współpracuje tylko z UUCP Taylora.

t
: Ten protokół jest przeznaczony do stosowania w połączeniach TCP lub innych sieciach naprawdę wolnych od błędów. Wykorzystuje 1024-bajtowe pakiety i wymaga połączenia 8-bitowego.

e
: Ten protokół w zasadzie działa tak samo jak *t*. Główna różnica polega na tym, że *e* jest protokołem strumieniowym i dlatego nadaje się jedynie do stosowania w niezawodnych sieciach.

f
: Ten protokół jest przeznaczony do stosowania w niezawodnych połączeniach X.25. Jest to protokół strumieniowy i wymaga 7-bitowej ścieżki danych. Znaki 8-bitowe są maskowane, co może powodować, że protokół nie będzie efektywny.

G
: Jest to wersja protokołu *g* implementowana przez 4. wydanie systemu V. Jest obsługiwany również przez inne wersje UUCP.

a
: Ten protokół jest podobny do protokołu ZMODEM. Wymaga połączenia 8-bitowego, ale maskuje pewne znaki sterujące, jak XON i XOFF.

Strojenie protokołu transmisji

Wszystkie protokoły tolerują pewne różnice w rozmiarach pakietów, czasach oczekiwania itp. Zwykle domyślne wartości działają dobrze w standardowych warunkach, ale mogą nie być optymalne w twojej sytuacji. Na przykład protokół *g* wyko-

rzystuje rozmiary okien od 1 do 7 i rozmiary pakietów będące potęgą 2 począwszy od 64 do 4096. Jeżeli twoja linia telefoniczna zwykle jest tak zakłócona, że gubi więcej niż 5 procent wszystkich pakietów, powinieneś zmniejszyć rozmiar pakietu i okna. Z drugiej strony przy bardzo dobrych liniach telefonicznych, potwierdzanie każdego 128-bajtowego pakietu może być marnotrawstwem, a więc może warto wtedy zwiększyć rozmiar pakietu do 512 lub nawet 1024 bajtów. Większość plików binarnych zawartych w dystybucjach Linuksa ma domyślną wartość rozmiaru okna 7 i pakietu 128 bajtów.

UUCP Taylora pozwala dostroić parametry poleceniem *protocol-parameter* umieszczanym w pliku *sys*. Na przykład, aby ustawić rozmiar pakietu dla protokołu *g* na wartość 512 w czasie połączenia z **pablo**, musisz dodać:

```
system      pablo
...
protocol-parameter g packet-size 512
```

Parametry, które można zmieniać, i ich nazwy nie są jednakowe dla wszystkich protokołów. Ich pełną listę znajdziesz w dokumentacji dołączonej do kodu źródłowego UUCP Taylora.

Wybór określonych protokołów

Nie każda implementacja *uucico* rozumie wszystkie protokoły, a więc w fazie wstępnego uzgadniania oba procesy muszą ustalić jeden wspólny protokół. Nadrzędny system *uucico* oferuje podległemu listę obsługiwanych protokołów, wysyłając `Pprotlist`, z której system podległy musi coś wybrać.

W oparciu o typ używanego portu (modem, TCP lub połączenie bezpośrednie), *uucico* tworzy domyślną listę protokołów. Dla modemu i połączenia bezpośredniego lista ta zwykle zawiera protokoły *i*, *a*, *g*, *G* i *j*. Dla połączeń TCP lista zawiera *t*, *e*, *i*, *a*, *g*, *G*, *j* i *f*. Taką domyślną listę wolno zmienić za pomocą polecenia *protocols*, które można umieścić w opisie zarówno systemu, jak i portu. Na przykład mógłbyś dokonać edycji pliku *port* twojego modemu i zmienić go tak:

```
port        serial1
...
protocols     igG
```

Tym sposobem wszelkie przychodzące i wychodzące przez ten port połączenia będą używały *i*, *g* lub *G*. Jeżeli system zdalny nie obsługuje żadnego z nich, połączenie się nie powiedzie.

Rozwiązywanie problemów

Ten podrozdział opisuje, co może się nie udać przy połączeniu UUCP, i podpowiada, gdzie szukać błędu i jak go poprawić. Pokazujemy tu najczęściej spotykane problemy, ale wystąpić mogą też inne – po prostu nie wszystkie zdołaliśmy tu pokazać.

Jeżeli masz problem, włącz debugowanie przez *–xall* i przyjrzyj się wynikowi w pliku *Debug* w katalogu bufora. Plik ten powinien ci pomóc szybko rozpoznać problem. Często pomocne jest włączenie głośnika modemu, gdy nie nawiązuje on połączenia.

W przypadku modemów kompatybilnych ze standardem Hayesa, możesz włączyć głośnik, dodając `ATL1M1 OK` w dialogu modemu w pliku *dial*.

Przede wszystkim powinieneś sprawdzić, czy wszystkie prawa dostępu do plików są poprawne. *uucico* powinno mieć prawo setuid **uucp**, a wszystkie pliki w katalogach */usr/lib/uucp*, */var/spool/uucp* i */var/spool/uucppublic* powinny być własnością **uucp**. Istnieje także kilka plików ukrytych w katalogu buforowym. One także muszą być własnością **uucp***.

Gdy jesteś już pewny, że prawa dostępu do wszystkich plików są poprawne, a nadal masz problemy, możesz zacząć bardziej dosłownie traktować komunikaty o błędach. Przyjrzyjmy się teraz kilku najczęściej spotykanym błędom i problemom.

uucico wciąż mówi „Wrong Time to Call"

Oznacza to, że prawdopodobnie w opisie systemu w pliku *sys* nie podałeś polecenia *time* ze szczegółami dotyczącymi dzwonienia lub zapisałeś je w taki sposób, że nie można ich wykorzystać. Jeżeli nie ma rozkładu dzwonienia, *uucico* uzna je, że do systemu nie można dzwonić.

uucico zgłasza, że ośrodek jest już zablokowany

Oznacza to, że *uucico* wykrywa w katalogu */var/spool/uucp* plik blokujący dla systemu zdalnego. Plik blokujący może być pozostałością po poprzednim połączeniu, które uległo awarii lub zostało przerwane. Inne wytłumaczenie to inny proces *uucico*, który próbował zadzwonić do systemu zdalnego i zatrzymał się w skrypcie dialogowym lub z innego powodu.

Aby naprawić ten błąd, za pomocą sygnału zawieszenia zatrzymaj wszystkie aktywne procesy *uucico* dla danego ośrodka i usuń wszelkie pozostałe po nich pliki blokujące.

Możesz podłączyć się do ośrodka zdalnego, ale skrypt dialogowy nie działa

Przyjrzyj się komunikatowi otrzymanemu z ośrodka zdalnego. Jeżeli jest on uszkodzony, może to oznaczać problemy z prędkością. W przeciwnym razie potwierdź, że naprawdę zgadza się z tym, czego oczekuje twój skrypt dialogowy. Pamiętaj, że skrypt dialogowy rozpoczyna od ciągu oczekiwanego. Jeżeli odebrałeś monit logowania i wysłałeś swoją nazwę, ale nigdy nie dostałeś pytania o hasło, wstaw opóźnienie przed jej wysłaniem lub nawet między literami. Może działasz za szybko dla swojego modemu.

Twój modem nie dzwoni

Jeżeli twój modem nie pokazuje, że linia DTR została uaktywniona, gdy *uucico* dzwonił, urządzenie *uucico* może nie być poprawne. Jeżeli twój modem rozpoznaje DTR, sprawdź programem terminala, że możesz pisać do modemu. Jeżeli to działa, włącz echo opcją `\E` na początku dialogu z modemem. Jeżeli modem nie powtórzy

* To znaczy pliki o nazwach rozpoczynających się od kropki. Takie pliki normalnie nie są wyświetlane przez polecenie *ls*.

twojego polecenia w czasie dialogu, sprawdź, czy prędkość linii jest odpowiednia. Jeżeli zobaczysz echo, sprawdź, czy wyłączyłeś odpowiedzi modemu lub ustawiłeś kody. Zweryfikuj poprawność samego skryptu dialogowego. Pamiętaj, że aby wysłać do modemu odwrotny ukośnik musisz napisać dwa takie nośniki.

Twój modem próbuje dzwonić, ale nie udaje mu się wyjść na zewnątrz

Wstaw opóźnienie w numerze telefonu, szczególnie jeżeli musisz wybierać specjalną sekwencję, aby wyjść z sieci telefonicznej firmy na zewnątrz. Upewnij się, że używasz poprawnego typu urządzenia, ponieważ niektóre sieci telefoniczne obsługują tylko jeden typ dzwonienia. Sprawdź dwa razy numer telefonu, by mieć pewność, że jest on poprawny.

Logowanie udaje się, ale uzgadnianie nie

Może to oznaczać wiele różnych problemów. Wynik pliku log powinien coś podpowiedzieć. Zobacz, jakie protokoły oferuje zdalny ośrodek (wysyła on w czasie uzgadniania ciąg znaków `P protlist`). Aby uzgadnianie się powiodło, obie strony muszą obsługiwać przynajmniej jeden wspólny protokół, a więc sprawdź, czy to robią.

Jeżeli system zdalny wysyła *RLCK*, oznacza to, że w ośrodku zdalnym, do którego jesteś już podłączony przez inną linię, zalega przedawniony plik blokujący. W takiej sytuacji poproś administratora systemu zdalnego o usunięcie pliku.

Jeżeli zdalny system wysyła *RBADSEQ*, to ma włączone zliczanie połączeń z tobą, ale licznik się nie zgadza. Jeżeli wysyła *RLOGIN*, nie masz prawa zalogować się z danym ID.

Pliki log i debugowanie

Gdy kompilujesz pakiet UUCP, by wykorzystywał Taylorowskie logowanie błędów, masz tylko trzy pliki log znajdujące się w katalogu buforowym. Główny plik log nosi nazwę *Log* i zawiera wszelkie informacje o zestawionych połączeniach i przesłanych plikach. Typowy fragment wygląda (po niewielkim przeformatowaniu w celu dopasowania do strony) tak:

```
uucico pablo - (1994-05-28 17:15:01.66 539) Calling system pablo (port cua3)
uucico pablo - (1994-05-28 17:15:39.25 539) Login successful
uucico pablo - (1994-05-28 17:15:39.90 539) Handshake successful
        (protocol 'g' packet size 1024 window 7)
uucico pablo postmaster (1994-05-28 17:15:43.65 539) Receiving D.pabloB04aj
uucico pablo postmaster (1994-05-28 17:15:46.51 539) Receiving X.pabloX04ai
uucico pablo postmaster (1994-05-28 17:15:48.91 539) Receiving D.pabloB04at
uucico pablo postmaster (1994-05-28 17:15:51.52 539) Receiving D.pabloX04as
uucico pablo postmaster (1994-05-28 17:15:54.01 539) Receiving D.pabloB04c2
uucico pablo postmaster (1994-05-28 17:15:57.17 539) Receiving D.pabloX04c1
uucico pablo - (1994-05-28 17:15:59.05 539) Protocol 'g' packets: sent 15,
               resent 0, received 32
uucico pablo - (1994-05-28 17:16:02.50 539) Cal complete (26 seconds)
uuxqt pablo postmaster (1994-05-28 17:16:11.41 546) Executing X.pabloX04ai
               (rmail okir)
uuxqt pablo postmaster (1994-05-28 17:16:13.30 546) Executing X.pabloX04as
               (rmail okir)
```

```
uuxqt pablo postmaster (1994-05-28 17:16:13.51 546) Executing X.pabloX04c1
              (rmail okir)
```

Drugi ważnym plikiem log jest *Stats*, który zawiera statystyki dotyczące przesyłanych plików. Część *Stats* odpowiedzialna za powyższe transfery wygląda tak (znów wiersze podzielono, aby zmieścić je na stronie):

```
postmaster pablo (1994-05-28 17:15:44.78)
   received 1714 bytes in 1.802 seconds (951 bytes/sec)
postmaster pablo (1994-05-28 17:15:46.66)
   received 57 bytes in 0.634 seconds (89 bytes/sec)
postmaster pablo (1994-05-28 17:15:49.91)
   received 1898 bytes in 1.599 seconds (1186 bytes/sec)
postmaster pablo (1994-05-28 17:15:51.67)
   received 65 bytes in 0.555 seconds (117 bytes/sec)
postmaster pablo (1994-05-28 17:15:55.71)
   received 3217 bytes in 2.254 seconds (1427 bytes/sec)
postmaster pablo (1994-05-28 17:15:57.31)
   received 64 bytes in 0.590 seconds (110 bytes/sec)
```

Trzeci plik to *Debug*. Zapisywane są w nim informacje pomocne w debugowaniu. Jeżeli używasz debugowania, upewnij się, czy plik ten ma prawa dostępu 600. W zależności od wybranego trybu debugowania może zawierać nazwę użytkownika i hasło używane do połączenia ze zdalnym systemem.

Jeżeli masz jakieś narzędzia do przeglądania plików log w tradycyjnym formacie używanym przez implementacje kompatybilne z HDB, możesz również skompilować UUCP Taylora tak, by generowało logi w stylu HDB. To kwestia włączenia opcji w pliku *config.h* w czasie kompilacji.

17

Poczta elektroniczna



Przesyłanie poczty elektronicznej pozostaje najbardziej widocznym zastosowaniem sieci, począwszy od jej wynalezienia. U zarania e-mail był prostą usługą, która polegała na kopiowaniu pliku z komputera na komputer i dodawaniu go do pliku *skrzynki pocztowej* odbiorcy. Idea wciąż jest ta sama, choć stale rozwijająca się sieć ze złożonymi zasadami rutingu i rosnącą liczbą wiadomości wymusiła powstanie bardziej skomplikowanych schematów działania.

Opracowano różne standardy wymiany poczty. Ośrodki w Internecie przyjęły standard wyłożony w RFC-822 i rozwijany w dalszych RFC. Jest to niezależny od komputera sposób na przesyłanie przez pocztę elektroniczną po prostu *wszystkiego*, włącznie z grafiką, plikami dźwiękowymi i zestawami znaków specjalnych*. CCITT zdefiniowała inny standard, X.400, który jeszcze funkcjonuje w dużych firmach i organizacjach rządowych, ale stopniowo wychodzi z użycia.

Dla systemów Unix stworzono całkiem sporą liczbę programów do przesyłania poczty. Jeden z najlepiej znanych to *sendmail*, napisany przez Erica Allmana z Uniwersytetu Kalifornijskiego w Berkeley. Eric Allman obecnie udostępnia *sendmail* w ramach komercyjnego przedsięwzięcia, ale program pozostaje darmowy. *sendmail* jest dostarczany jako standardowy agent pocztowy w wielu dystrybucjach Linuksa. Konfigurację *sendmaila* opisujemy w rozdziale 18, *Sendmail*.

Linux wykorzystuje również Exima, napisanego przez Philipa Hazela z uniwersytetu w Cambridge. Konfigurację Exima opisujemy w rozdziale 19, *Exim*.

W porównaniu z *sendmailem*, *Exim* ma raczej skromne możliwości. Wielu ośrodkom potrzebującym poczty jednak wystarczają.

Zarówno Exim, jak i *sendmail* obsługują zestaw plików konfiguracyjnych, który musi być dostosowany do potrzeb systemu. Poza informacjami, których wymaga podsystem poczty (jak nazwa hosta lokalnego), istnieje wiele parametrów, które można do-

* Jeżeli nie wierzysz, przeczytaj RFC-1437.

stosowywać. Przy pierwszym zetknięciu główny plik konfiguracyjny *sendmaila* jest bardzo trudny do zrozumienia. Wygląda jakby twój kot zdrzemnął się na klawiaturze, nacisnąwszy klawisz [Shift]. Pliki konfiguracyjne Exima są bardziej uporządkowane i łatwiejsze do zrozumienia. Jednak Exim nie obsługuje bezpośrednio UUCP, lecz tylko adresy domenowe. Może teraz nie jest to już tak niedogodne, jak niegdyś. Większości ośrodków ograniczenia Exima nie przeszkadzają. Jednak konfiguracja obu programów jest równie czasochłonna.

W tym rozdziale powiemy, co to jest poczta elektroniczna i z jakimi zagadnieniami styka się jej administrator. Rozdziały 18 i 19 zawierają instrukcje dotyczące konfiguracji *sendmaila* i Exima. Powinny one wystarczyć do uruchomienia mniejszych ośrodków, ale oczywiście opcji jest dużo więcej i możesz spędzić wiele godzin przed swoim komputerem na konfigurowaniu wymyślnych funkcji.

W tym rozdziale krótko omówimy ustawienie *elma* – popularnego agenta pocztowego użytkownika dla systemów uniksowych, także dla Linuksa.

Więcej informacji na temat poczty elektronicznej w Linuksie znajdziesz w *Electronic Mail HOWTO*, autorstwa Guylhema Aznara*; ten dokument jest regularnie rozsyłany na listę dyskusyjną *comp.os.linux.answers*. Pakiety dystrybucyjne *elma*, *Exima* i *sendmaila* także zawierają szczegółowe dokumentacje, które powinny odpowiedzieć na większość pytań na temat ich konfigurowania, a my w odpowiednich rozdziałach podajemy odniesienia do tej dokumentacji. Jeżeli potrzebujesz ogólnych informacji na temat poczty elektronicznej, przejrzyj różne RFC.

Co to jest wiadomość pocztowa

Mówiąc bardzo ogólnie, wiadomość pocztowa składa się z treści i specjalnych danych administracyjnych określających odbiorcę, sposób przesyłania, i tym podobne informacje, które również widzisz, gdy patrzysz na normalną kopertę listu.

Te dane administracyjne pasują do dwóch kategorii. W pierwszej znajdują się wszelkie dane, właściwe dla sposobu przesyłania, jak adres nadawcy i odbiorcy. Dla tego nazywa się je *kopertą*. Można je zmieniać przez oprogramowanie transportowe w czasie przekazywania wiadomości.

Druga kategoria to wszelkie dane niezbędne do obsłużenia wiadomości, niezwiązane z żadnym szczególnym mechanizmem transportowym, czy li temat wiadomości, lista wszystkich odbiorców i data wysłania wiadomości. W wielu sieciach przyjęło się poprzedzanie wiadomości tymi danymi, a więc utworzenie tak zwanego *nagłówka poczty*. Jest on oddzielony pustym wierszem od *treści wiadomości***.

Większość oprogramowania do przesyłania poczty w świecie Uniksa używa formatu nagłówka zdefiniowanego w RFC-822. Pierwotnym celem tego dokumentu było określenie standardu dla sieci ARPANET, ale ponieważ z założenia był to stan-

* Z Guylhemem można się skontaktować pod adresem *guylhem@danmark.linux.eu.org*.

** Do klienta należy decyzja o dołączaniu do wiadomości pliku *signature* lub *.sig*, zwykle zawierającego informacje o autorze wraz z żartem lub mottem. Jest on oddzielony od treści wiadomości wierszem zawierającym znak – i spację.

dard niezależny od środowiska, łatwo został zaadaptowany do innych sieci, włącznie z wieloma sieciami opartymi na UUCP.

Jednak RFC-822 jest najmniejszym wspólnym mianownikiem. W ostatnich latach wymyślono nowe standardy, aby poradzić sobie z nowymi potrzebami, takimi jak szyfrowanie danych, międzynarodowy zestaw znaków i MIME (*Multipurpose Internet Mail Extensions*), opisany m.in. w RFC-1341.

We wszystkich tych standardach nagłówek składa się z kilku wierszy oddzielonych sekwencją końca wiersza. Wiersz składa się z nazwy pola w pierwszej kolumnie i samego pola oddzielonego dwukropkiem i białym znakiem. Format i składnia każdego pola zależą od nazwy pola. Pole nagłówka można przenosić do następnego nowego wiersza, jeżeli rozpoczyna się on od białego znaku, np. tabulatora. Pola mogą pojawiać się w dowolnej kolejności.

Typowy nagłówek poczty może wyglądać tak:

```
Return-Path: <ph10@cus.cam.ac.uk>
Received: ursa.cus.cam.ac.uk (cusexim@ursa.cus.cam.ac.uk [131.111.8.6])
   by al.animats.net (8.9.3/8.9.3/Debian 8.9.3-6) with ESMTP id WAA04654
   for <terry@animats.net>; Sun, 30 Jan 2000 22:30:01 +1100
Received: from ph10 (helo=localhost) by ursa.cus.cam.ac.uk with local-smtp
   (Exim 3.13 #1) id 12EsYC-0001EeF-00; Sun, 30 Jan 2000 11:29:52 +0000
Date: Sun, 30 Jan 2000 11:29:52 +0000 (GMT)
From: Philip Hazel <ph10@cus.cam.ac.uk>
Reply-To: Philip Hazel <ph10@cus.cam.ac.uk>
To: Terry Dawson <terry@animats.net>, Andy Oram <andyo@oreilly.com>
Subject: Electronic mail chapter
In-Reply-To: <38921283.A58948F2@animats.net>
Message-ID: <Pine.SQL.3.96.1000130111515.5800A-200000@ursa.cus.cam.ac.uk>
```

Zwykle wszystkie wymagane pola nagłówka są generowane przez interfejs programu pocztowego, takiego jak *elm*, *pine*, *mush* czy *mailx*. Jednak niektóre są opcjonalne i mogą być dodawane przez użytkownika. Na przykład *elm* pozwala edytować część nagłówka poczty. Pozostałe są dodawane przez oprogramowanie przesyłające pocztę. Jeżeli zajrzysz do pliku lokalnej skrzynki pocztowej, możesz zobaczyć, że każda wiadomość jest poprzedzona linią „From" (uwaga: bez dwukropka). *Nie* jest to nagłówek RFC-822. Został on wstawiony przez twój program pocztowy dla wygody programu czytającego wiadomości z twojej skrzynki. Aby uniknąć potencjalnych problemów z wierszami w treści wiadomości, które rozpoczynają się również od słowa „From", standardową procedurą jest maskowanie każdego takiego wystąpienia znakiem >.

Oto zbiór popularnych pól nagłówków i ich znaczenie:

`From:`
: Zawiera adres pocztowy nadawcy i czasem także jego „prawdziwe nazwisko". Formatowane bardzo różnie.

`To:`
: Jest to lista adresatów wiadomości. Lista adresów jest oddzielana przecinkami.

Cc:
: Jest to lista adresów e-mail, na które zostanie wysłana kopia „do wiadomości". Lista adresów jest oddzielana przecinkami.

`Bcc:`
: Jest to lista adresów e-mail, na które zostanie wysłana kopia „do wiadomości". Kluczowa różnica pomiędzy „Cc:" i „Bcc:" jest taka, że adresy wpisane w „Bcc:" nie pojawiają się w nagłówku dostarczonej wiadomości u żadnego odbiorcy. Jest to sposób na powiadomienie adresatów, że wysłałeś kopie wiadomości do innych osób, bez mówienia do kogo. Lista adresów jest oddzielana przecinkami.

`Subject:`
: W kilku słowach opisuje zawartość poczty.

`Date:`
: Zawiera datę i czas wysłania wiadomości.

`Reply-To:`
: Określa adres, na który nadawca chce przekierować odpowiedź odbiorcy. Może być to przydatne, jeżeli masz kilka kont, ale chcesz otrzymywać masową pocztę na tym, którego używasz najczęściej. To pole jest opcjonalne.

`Organization:`
: Organizacja będąca właścicielem komputera, z którego pochodzi wiadomość. Jeżeli twój komputer jest własnością prywatną, pozostaw to pole puste lub wstaw słowo „private" albo coś zupełnie bez sensu. To pole nie jest opisane w RFC i jest całkowicie opcjonalne. Niektóre programy pocztowe obsługują je bezpośrednio, inne tego nie robią.

`Message-ID:`
: Ciąg znaków wygenerowany przez program do przesyłania poczty w systemie wyjściowym. Jest to unikalny identyfikator wiadomości.

`Received:`
: Każdy ośrodek, który przetwarza twoją wiadomość (włącznie z maszynami nadawcy i odbiorcy), wstawia takie pole do nagłówka, podając nazwę ośrodka, ID wiadomości, czas i datę odebrania, z jakiego ośrodka wiadomość nadeszła i jakie oprogramowanie transportowe zostało użyte. Te linie pozwalają ci prześledzić trasę poczty i zgłosić reklamację do odpowiedzialnej osoby, jeżeli coś poszło nie tak.

`X-`*`cokolwiek`*`:`
: Żaden program związany z pocztą nie powinien narzekać na nagłówek, który rozpoczyna się od `X-`. Jest on używany do implementacji dodatkowych funkcji, które nie zostały jeszcze uwzględnione w RFC lub nigdy nie będą. Na przykład istniał kiedyś bardzo duży serwer list pocztowych dla Linuksa, który pozwalał określić, z jakim kanałem chcesz się połączyć, przez wstawienie ciągu znaków **`X-Mn-Key`**`:` i nazwy kanału.

Jak jest dostarczana poczta

W zasadzie pocztę będziesz pisał, używając interfejsu pocztowego, takiego jak *mail* czy *mailx*, albo bardziej wyrafinowanych programów, takich jak *mutt*, *tkrat* czy *pine*. Programy te są nazywane *agentami pocztowymi użytkownika* (ang. *mail user agents*), w skrócie: MUA. Gdy wydajesz polecenie wysłania wiadomości, program interfejsu w większości przypadków przekazuje ją innemu programowi – doręczycielowi. Są to tak zwane *agenty przesyłania wiadomości* (ang. *mail transport agents*), w skrócie: MTA. W większości systemów to samo MTA jest używane do dostarczania zarówno poczty lokalnej, jak i zdalnej. Zwykle jest wywoływane jako */usr/sbin/sendmail* lub jako */usr/lib/sendmail* w systemach niezgodnych z FSSTND. W systemach UUCP nie jest niczym niezwykłym fakt, że dostarczanie jest obsługiwane przez dwa oddzielne programy: *rmail* dla poczty zdalnej i *lmail* dla poczty lokalnej.

Lokalne dostarczanie poczty jest oczywiście czymś więcej niż dołączeniem przychodzącej wiadomości do skrzynki pocztowej odbiorcy. Zwykle lokalne MTA rozumie aliasy (konfigurowane po to, aby adresy odbiorcy wskazywały na inne adresy) i przekazywanie (przekierowywanie poczty użytkownika w inne miejsce). Poza tym wiadomości, które nie mogą zostać dostarczone, zwykle muszą być *odbite* (ang. *bounced*), to znaczy zwrócone do nadawcy wraz z informacją o błędzie.

W przypadku dostarczania zdalnego, oprogramowanie transportowe zależy od typu połączenia. Poczta wędrująca przez sieć TCP/IP zwykle używa protokołu SMTP (*Simple Mail Transfer Protocol*), który jest opisany w RFC-821. SMTP ma dostarczać pocztę bezpośrednio do komputera odbiorcy, negocjując transfer wiadomości z demonem SMTP strony zdalnej. Obecnie obsługę poczty w firmach organizuje się w ten sposób, że tworzy się hosty, które przejmują całą pocztę dla adresatów z firmy, a następnie kierują ją do wskazanego odbiorcy.

W sieciach UUCP poczta zwykle nie jest dostarczana bezpośrednio, lecz jest przekazywana na do hosta docelowego przez szereg systemów pośrednich. Aby wysłać wiadomość przez łącze UUCP, MTA po stronie nadawcy zwykle wykonuje polecenie *rmail* na systemie przekazującym, używając w tym celu *uux*, i przekazuje wiadomość na standardowe wejście.

Ponieważ *uux* jest wywoływane oddzielnie dla każdej wiadomości, może znacznie obciążać główne huby pocztowe oraz zaśmiecać kolejki buforowe UUCP setkami małych plików zajmujących nieproporcjonalnie wiele miejsca na dysku*. Dlatego niektóre MTA pozwalają ci zbierać kilka wiadomości dla systemu zdalnego w jeden plik wsadowy. Plik wsadowy zawiera też polecenia SMTP, które zwykle wydaje host lokalny, jeżeli było użyte bezpośrednie połączenie SMTP. Nazywa się to *wsadowe* SMTP lub BSMTP. Dalej plik wsadowy jest przekazywany do programu *rsmtp* lub *bsmtp* w systemie zdalnym, który przetwarza dane wejściowe prawie tak samo, jak by to było normalne połączenie SMTP.

* To dlatego, że przestrzeń dyskowa jest zwykle alokowana w blokach po 1024 bajty. A więc nawet kilkubajtowa wiadomość zajmuje pełny kilobajt.

Adresy e-mail

Adresy e-mail składają się przynajmniej z dwóch części. Jedna część to nazwa *domeny pocztowej*, która ostatecznie zostanie przetłumaczona na host adresata lub jakiś host przyjmujący pocztę w jego imieniu. Druga część to unikalna nazwa użytkownika. Może to być jego nazwa logowania, rzeczywiste nazwisko, postać „Imię Nazwisko” albo dowolny alias, który zostanie przetłumaczony na nazwę użytkownika lub listę użytkowników. Inne schematy adresowania, jak X.400, używają bardziej ogólnego zestawu „atrybutów”, używanych do poszukiwania hosta adresata na serwerze usług katalogowych X.500.

To, jak adresy e-mail są interpretowane, zależy w dużym stopniu od typu sieci. Nas interesuje, jak sieci TCP/IP i UUCP interpretują adresy e-mail.

RFC-822

Ośrodki internetowe stosują standard RFC-822. Jest to dobrze znany wszystkim zapis: *nazwa_użytkownika@host.domena*, gdzie *host.domena* to pełna nazwa domenowa hosta. Poprawna nazwa znaku oddzielającego te dwie części to „commercial at”, ale wygodniej czytać go jako „at”. Taki zapis nie określa trasy prowadzącej do hosta docelowego. Ruting wiadomości jest obsługiwany przez mechanizmy, które opiszemy wkrótce.

Z RFC-822 będziesz się stale spotykał, jeżeli masz ośrodek podłączony do Internetu. Jego zastosowanie wychodzi poza pocztę i obejmuje też inne usługi, takie jak grupy dyskusyjne. To, jak RFC-822 jest używane w grupach dyskusyjnych, omawiamy w rozdziale 20, *Grupy dyskusyjne*.

Dawne formaty pocztowe

W pierwotnym środowisku UUCP powszechnie stosowana była następująca postać adresu: *ścieżka!host!użytkownik*, gdzie *ścieżka* opisywała kolejność hostów, przez które wiadomość musiała przejść, aby osiągnąć *host* docelowy. Ta konstrukcja jest nazywana *wykazem trasowania* (ma też zwyczajową nazwę angielską *bang path*, od potocznej nazwy wykrzyknika *bang*). Obecnie wiele sieci opartych na UUCP stosuje się do RFC-822 i rozumie adresy oparte na domenach.

Inne sieci wciąż inaczej rozumieją adresowanie. Na przykład sieci oparte na DECnet wykorzystują dwa dwukropki jako separator w adresie o postaci *host::użytkownik**. Standard X.400 wykorzystuje zupełnie inny schemat, opisując odbiorcę parą atrybut-wartość, jak np. kraj i firma.

W sieci FidoNet każdy użytkownik jest identyfikowany przez kod typu **2:320/204.9** składający się z czterech liczb oznaczających strefę (2 to Europa), sieć (320 to Paryż i okolice), węzeł (lokalny hub) i punkt (PC indywidualnego użytkownika). Adresy FidoNet mogą być odwzorowane na adresy standardu RFC-822. Przykład powyż-

* Jeżeli próbujesz dotrzeć do adresu DECnetu ze środowiska RFC-822, możesz użyć zapisu *„host::użytkownik”@przekaźnik*, gdzie *przekaźnik* to nazwa znanego przekaźnika pomiędzy Internetem a DECnetem.

szy mógłby zostać zapisany jako *Thomas.Quinot@p9.f204.n320.z2.fidonet.org*. Nie musimy dodawać, że nazwy domen były łatwe do zapamiętania.

Łączenie różnych formatów poczty

Wiadomo, że tam, gdzie spotyka się kilka różnych standardów i paru mądrych ludzi, będą oni szukać sposobu na połączenie różnych systemów, tak by mogły ze sobą współpracować. W rezultacie istnieje szereg gatewayów, które łączą ze sobą ze dwa różne systemy pocztowe, tak że poczta może być przekazywana z jednego do drugiego. Przy łączeniu dwóch systemów krytycznym problemem jest adresowanie. Nie będziemy szczegółowo rozważać samych gatewayów, ale przyjrzymy się kilku komplikacjom w adresowaniu, które mogą wystąpić przy pewnego typu gatewayach.

Zajmijmy się połączeniem dwóch różnych zapisów adresów: wykazem trasowania UUCP i RFC-822. Te dwa typy adresowania nie współpracują zbyt dobrze. Załóżmy, że mamy adres *domenaA!użytkownik@domenaB*. Nie jest jasne, czy znak @ ma wyższy priorytet niż ścieżka, czy odwrotnie: czy mamy wysłać wiadomość do *domenyB* i użytkownika *domenaA!użytkownik*, czy może do *domenyA*, która przekaże wiadomość do *użytkownika* w *domenieB*.

Adresy łączące różne typy operatorów są nazywane *adresami hybrydowymi*. Właśnie pokazany, najpowszechniejszy typ adresu jest zwykle rozwiązywany przez nadanie priorytetu znakowi @, a nie ścieżce. W przypadku *domenaA!użytkownik@domenaB*, oznacza to wysłanie wiadomości najpierw do *domenyB*.

Jednak istnieje sposób na określenie tras w RFC-822: <*@domenaA,@domenaB:użytkownik@domenaC*> oznacza adres *użytkownika* w *domenieC*, gdzie *domenaC* jest osiągalna przez *domenęA* i *domenęB* (w tej kolejności). Ten typ adresu często jest nazywany adresem *rutowanym źródłowo* (ang. *source routed*). Nie należy jednak polegać na jego działaniu, gdyż wersje RFC opisujące ruting poczty zalecają, by ruting źródłowy w adresie poczty był ignorowany i by była podejmowana próba bezpośredniego dostarczenia wiadomości do zdalnego celu.

W przypadku adresu z operatorem %: *użytkownik%domenaB@domenaA*, poczta najpierw jest wysyłana do *domenyA*, a znak % jest zamieniany na @. Adres w tym momencie ma postać **użytkownik@domenaB** i program pocztowy przekazuje twoją wiadomość do *domenyB*, w której jest dostarczana do podanego *użytkownika*. Ten typ adresu jest czasem nazywany „Ye Olde ARPAnet Kludge". Nie radzimy go używać.

Czasami istnieją pewne wskazania do używania różnego sposobu adresowania. Zostaną one opisane w kolejnych podrozdziałach. W środowisku RFC-822 zalecamy adresy bezwzględne postaci *użytkownik@host.domena*; raczej trzeba unikać innych formatów.

Jak działa ruting poczty

Proces przekierowujący wiadomość do hosta adresata jest nazywany *rutingiem*. Poza znalezieniem ścieżki od nadawcy do odbiorcy, uwzględnia sprawdzanie błędów i może też uwzględniać prędkość i optymalizację kosztów.

Istnieje duża różnica pomiędzy sposobem, w jaki ośrodek UUCP obsługuje ruting, a sposobem, w jaki robi to ośrodek internetowy. W Internecie główne zadanie kierowania danych do hosta adresata (gdy już jest znany jego adres IP) jest realizowane przez warstwę sieciową IP, natomiast w strefie UUCP trasa musi być dostarczona przez użytkownika lub wygenerowana przez agenta przesyłającego pocztę.

Ruting poczty w Internecie

W Internecie konfiguracja hosta docelowego określa, czy jest realizowany jakiś szczególny ruting poczty. Domyślnie wiadomość jest dostarczana do celu następująco: stwierdza się, do jakiego hosta ma być wysłana i przekazuje mu bezpośrednio. Większość ośrodków internetowych chce przekierowywać całą przychodzącą pocztę do serwera pocztowego, który jest stale dostępny, i jest w stanie obsłużyć cały ruch i rozesłać pocztę lokalnie. Aby rozgłosić tę usługę, ośrodek rozdaje przez bazę DNS tak zwane rekordy MX dla swojej lokalnej domeny. Skrót MX pochodzi od *Mail Exchanger* (systemwymieniający pocztę); termin ten oznacza, że serwer działa jako system przekazujący pocztę dla wszystkich adresów z danej domeny. Rekordy MX mogą być używane również do obsługi ruchu na rzecz hostów, które same nie są podłączone do Internetu, jak sieci UUCP czy hosty FidoNet, których poczta musi być przekazywana przez gateway.

Rekordom MX zawsze jest przypisywany jakiś *priorytet*. Jest to dodatnia liczba całkowita. Jeżeli istnieje kilka systemów wymieniających pocztę dla jednego hosta, agent transportowy będzie próbował wysłać wiadomość do hosta wymieniające go pocztę, mające go najniższy priorytet. Jeżeli mu się to nie uda, spróbuje użyć hosta z wyższą wartością. Jeżeli sam host lokalny jest systemem wymieniającym pocztę dla adresu docelowego, może przekazywać wiadomości tylko do hostów MX o niższym priorytecie niż jego własny. Jest to dobry sposób na uniknięcie pętli. Jeżeli nie istnieje rekord MX dla domeny lub nie pozostał żaden odpowiedni rekord, agent transportowy ma prawo sprawdzić, czy domena ma związany z nią adres IP, i próbuje dostarczyć pocztę bezpośrednio do tego hosta.

Załóżmy, że firma Foobar, Inc. chce, żeby jej poczta była obsługiwana przez ich komputer **mailhub**. W DNS-ie będzie miała następujące rekordy MX:

```
green.foobar.com    IN MX   5 mailhub.foobar.com.
```

Dzięki temu wiadomo, że **mailhub.foobar.com** jest systemem wymieniającym pocztę dla hosta **green.foobar.com** i ma priorytet 5. Host, który chce dostarczyć pocztę do *joe@green.foobar.com*, sprawdza DNS i znajduje rekord MX wskazujący na **mailhub**. Jeżeli nie ma rekordu MX o priorytecie mniejszym niż 5, wiadomość jest dostarczana do **mailhub**, który z kolei przekazuje ją do **green**.

Jest to tylko bardzo prosty przykład działania rekordów MX. Po więcej informacji na temat rutowania poczty zajrzyj do RFC-821, RFC-974 i RFC-1123 w Internecie.

Ruting poczty w świecie UUCP

Ruting poczty w sieciach UUCP jest dużo bardziej skomplikowany niż w Internecie, ponieważ oprogramowanie transportowe nie realizuje go samodzielnie. Kiedyś cała poczta była adresowana za pomocą wykazów trasowania. Wykazy trasowania wymieniały hosty, przez które należało przekazywać wiadomość; poszczególne hosty oddzielano wykrzyknikami, a za hostem docelowym podawano nazwę użytkownika. Aby zaadresować list do użytkownika Janet na komputerze **moria**, użyłbyś ścieżki *eek!swim!moria!janet*. Wiadomość zostałaby wysłana z twojego hosta do komputera **eek**, a stamtąd do **swim** i ostatecznie do **moria**.

Oczywistą wadą tej techniki jest to, że wymaga ona od ciebie pamiętania wielu rzeczy na temat topologii sieci, szybkich łączy itp., czego nie wymaga ruting w Internecie. Co gorsza, jeśli przeoczysz jakąś zmianę w topologii sieci – jak usunięte łącza lub hosty – wiadomość nie dojdzie. Gdy byś zaś przeniósł się w inne miejsce, z całą pewnością musiałbyś uaktualnić te wszystkie trasy.

Ruting źródłowy ma swoje uzasadnienie, jeśli istnieją dwuznaczne nazwy hostów. Na przykład załóżmy, że są dwa ośrodki o nazwie **moria**, jeden w Stanach Zjednoczonych, a drugi we Francji. Do którego z nich odnosi się adres *moria!janet*? Staje się to jednoznaczne dopiero wtedy, gdy określisz drogę, którą można dotrzeć do **moria**.

Pierwszym krokiem do unikalności nazw hostów było rozpoczęcie projektu mapowania UUCP. Jest on prowadzony w Rutgers University. Rejestruje się wszystkie oficjalne nazwy hostów UUCP wraz z informacją o ich sąsiadach UUCP i ich lokalizacji geograficznej. Informacje zebrane w ramach projektu mapowania UUCP są publikowane jako *Mapy Usenetu*, które z kolei są regularnie rozpowszechniane przez Usenet. Typowy opis systemu w mapie (po usunięciu komentarzy) wygląda tak*:

```
moria
   bert(DAILY/2),
   swim(WEEKLY)
```

Ten wpis mówi, że **moria** ma połączenie z **bertem**, z którym łączy się dwa razy dziennie, i ze **swimem**, z którym łączy się raz w tygodniu. Format pliku map omówimy za chwilę bardziej szczegółowo.

Korzystając z informacji zawartych w mapach, możesz automatycznie generować pełne ścieżki z twojego hosta do ośrodka docelowego. Ta informacja zwykle jest zapisywana w pliku *paths*, zwanym także *bazą danych aliasów ścieżek*. Załóżmy, że z map wynika, że możesz dotrzeć do **berta** przez **ernie**. Alias ścieżki dla hosta **moria** wygenerowany na podstawie poprzedniej szczątkowej mapy może wyglądać jakoś tak:

```
moria      ernie!bert!moria!%s
```

---

* Mapy dla ośrodków zarejestrowanych w projekcie mapowania UUCP są rozpowszechniane przez grupę dyskusyjną *comp.mail.maps*. Inne firmy mogą publikować oddzielne mapy dla własnych sieci.

Jeżeli teraz podasz adres docelowy *janet@moria.uucp*, twój MTA wykorzysta powyższą trasę i wyśle wiadomość do **ernie** z adresem postaci *bert!moria!janet*.

Tworzenie pliku *paths* na podstawie pełnych map Usenetu nie jest jednak dobrym pomysłem. Informacja w nich zawarta zwykle bywa zniekształcona, a czasami nieaktualna. Dlatego tylko kilka głównych hostów używa pełnych światowych map UUCP do tworzenia swoich plików *paths*. Większość ośrodków utrzymuje informacje o rutingu jedynie dla ośrodków z ich sąsiedztwa, a pocztę przeznaczoną dla ośrodków, których nie mogą znaleźć w swoich bazach, wysyłają do mądrzejszych hostów, które mają pełniejszą informację o rutingu. Ten schemat nazywa się *rutingiem do inteligentnych hostów* (ang. *smart-host routing*). Hosty, które mają tylko jedno łącze UUCP (tak zwane *ośrodki brzegowe*), same nie realizują żadnego rutingu. W pełni polegają na mądrym hoście.

Łączenie UUCP i RFC-822

Jak dotąd najlepszym lekarstwem na problemy rutowania poczty w sieciach UUCP jest przyjęcie systemu nazw domen w sieciach UUCP. Oczywiście, nie możesz przez UUCP zadawać zapytań do serwera nazw. Mimo to wiele ośrodków UUCP stworzyło małe domeny, które wewnętrznie koordynują ich ruting. Domeny te ogłaszają w mapach jeden lub dwa hosty jako swoje gateway'e, dlatego nie każdy host musi mieć wpis w domenie. Gateway'e obsługują całą pocztę, która przychodzi do domeny i z niej wychodzi. Schemat rutingu wewnątrz domeny jest zupełnie niewidoczny dla świata zewnętrznego.

Działa to bardzo dobrze w schemacie rutingu z inteligentnymi hostami. Globalne informacje o rutingu są utrzymywane jedynie przez gateway'e. Mniejsze hosty w domenie mają jedynie małe, ręcznie pisane pliki *paths*, które podają informacje o trasach w obrębie domeny i trasę do huba pocztowego. Nawet gateway'e pocztowe nie potrzebują już informacji o rutingu dla każdego hosta UUCP na świecie. Poza pełną informacją o obsługiwanej domenie, muszą one posiadać teraz w swojej bazie jedynie trasy do całych domen. Na przykład te aliasy ścieżek kierują całą pocztą dla ośrodków domeny **sub.org** do hosta **smurf**:

```
.sub.org        swim!smurf!%s
```

Poczta adresowana do *claire@jones.sub.org* będzie wysyłana do **swim** z adresem *smurf!jones!claire*.

Hierarchiczny porządek przestrzeni nazw domen pozwala serwerom pocztowym na łączenie dokładniejszych tras z ogólnymi. Na przykład system we Francji może mieć specjalne trasy dla poddomeny **fr**, ale całą pocztę dla hostów w domenie **us** kierować przez jakiś system w Stanach Zjednoczonych. W ten sposób ruting oparty na domenach (bo tak się ta technika nazywa) znacznie redukuje rozmiar bazy danych o rutingu oraz operacje administracyjne.

Główną korzyścią płynącą z używania nazw domenowych w środowisku UUCP jest jednak zgodność z RFC-822, co pozwala łatwo przekazywać pocztę pomiędzy sieciami UUCP a Internetem. Wiele obecnych domen UUCP ma połączenie z gateway'em internetowym, który działa jako ich inteligentny host. Wysłanie wiadomości przez

Internet jest szybsze, a informacje o rutingu są dokładniejsze, ponieważ hosty w Internecie posługują się DNS-em zamiast map Usenet.

Gateway internetowy domeny opartej na UUCP, który chce być dostępny z Internetu, ogłasza swój rekord MX (rekordy MX opisaliśmy wcześniej w tym podrozdziale, w sekcji *Ruting poczty w Internecie*). Załóżmy, że host **moria** należy do domeny **orcnet.org**. Na to miast **gcc.groucho.edu** działa jako jej gateway internetowy. **moria** używa więc **gcc2** jako inteligentnego hosta, a więc cała poczta dla obcych domen jest wysyłana przez Internet. Z drugiej strony **gcc2** ogłasza rekord MX dla domeny ***.orcnet.org** i dostarcza całą przychodzącą pocztę dla ośrodków **orcnet** do hosta **moria**. Gwiazdka w ***.orcnet.org** jest znakiem uniwersalnym; oznacza, że pasują tu wszystkie hosty z domeny nieposiadające rekordów MX. Tak zwykle dzieje się w przypadku czystych domen UUCP.

Pozostał nam do rozwiązania jeszcze jeden problem, a mianowicie: programy transportowe UUCP nie mogą obsługiwać pełnych nazw domenowych. Większość pakietów UUCP została tak zaprojektowana, by radzić sobie z nazwami hostów o długości do ośmiu znaków (a czasem nawet krótszymi), ale zupełnie nie są obsługiwane znaki niealfanumeryczne, takie jak kropki.

Dlatego trzeba tłumaczyć nazwy RFC-822 na nazwy hostów UUCP. To mapowanie jest zupełnie niezależne od implementacji. Jednym z powszechnie stosowanych sposobów odwzorowania pełnych nazw domenowych na nazwy UUCP jest zastosowanie pliku aliasów ścieżek:

```
moria.orcnet.org   ernie!bert!moria!%s
```

Dzięki temu, na podstawie pełnego adresu domenowego, zostanie wygenerowany czysty adres UUCP w postaci wykazu trasowania. Niektóre programy pocztowe robią to za pomocą specjalnego programu. Na przykład *sendmail* wykorzystuje *uucpxtable*.

Przekształcenie odwrotne (potocznie nazywane *domenizowaniem*) jest czasami wymagane przy wysyłaniu poczty z sieci UUCP do Internetu. Dopóki nadawca poczty używa w adresie docelowym pełnej nazwy domenowej, wystarczy pozostawić nazwę domeny w adresie na koperci e przy przekazywaniu wiadomości do inteligentnego hosta. Niektóre ośrodki UUCP jednak nie należą do domeny. Są one zwykle „domenizowane" przez dodanie pseudodomeny **uucp**.

Baza aliasów ścieżek zapewnia główne informacje o rutingu w sieciach UUCP. Typowy wpis wygląda tak (nazwa ośrodka i ścieżka są oddzielone tabulatorami):

```
moria.orcnet.org   ernie!bert!moria!%s
moria              ernie!bert!moria!%s
```

Na podstawie tego wpisu każda wiadomość przeznaczona dla hosta **moria** jest dostarczana przez **ernie** i **bert**. Jeżeli program nie ma niezależnego sposobu na odwzorowanie pomiędzy przestrzeniami nazw, musi zostać podana pełna nazwa domenowa hosta **moria** oraz jego nazwa UUCP.

Gdybyś chciał przekierować do przekaźnika poczty wszystkie wiadomości przeznaczone dla hostów znajdujących się wewnątrz tej domeny, mógłbyś podać także

ścieżkę w bazie aliasów ścieżek, poprzedzając nazwę domeny kropką oznaczającą cel. Na przykład, gdyby wszystkie hosty z domeny **sub.org** były osiągalne przez **swim!smurf**, wpis w bazie aliasów ścieżek mógłby wyglądać tak:

```
.sub.org        swim!smurf!%s
```

Pisanie plików z aliasami ścieżek jest dopuszczalne jedynie wtedy, gdy masz ośrodek, który nie musi zbyt wiele rutować. Jeżeli musisz realizować ruting dla wielu hostów, lepiej jest użyć polecenia *pathalias* do stworzenia pliku na podstawie plików map. Mapy mogą być utrzymywane w dużo prostszy sposób, ponieważ możesz po prostu dodawać lub usuwać system, edytując wpis i tworząc od nowa plik map. Choć mapy publikowane w ramach projektu mapowania Usenetu rzadko są używane do rutingu, mniejsze sieci UUCP mogą udostępniać informacje o rutingu w swoich własnych zestawach map.

Na plik map składa się przede wszystkim lista ośrodków odpytywanych przez każdy system lub tych, przez które jest odpytywany nasz system. Nazwa systemu rozpoczyna się w pierwszej kolumnie, a za nią następuje lista połączeń oddzielonych przecinkami. Lista może ciągnąć się przez kilka wierszy, jeżeli kolejne wiersze zaczynają się od tabulatora. Każde połączenie jest opisane przez nazwę ośrodka oraz koszt podany w nawiasach kwadratowych. Koszt to wyrażenie arytmetyczne złożone z liczb oraz wyrażeń symbolicznych, takich jak DAILY lub WEEKLY. Wiersze rozpoczynające się znakiem hasha są ignorowane.

Jako przykład rozważmy system **moria**, który odpytuje **swim.twobirds.com** dwa razy dziennie, a **bert.sesame.com** raz w tygodniu. Łącze do **bert** wykorzystuje wolny modem (2400 b/s). **moria** opublikuje następujące wpisy w mapach:

```
moria.orcnet.org
   bert.sesame.com(DAILY/2),
   swim.twobirds.com(WEEKLY+LOW)
moria.orcnet.org = moria
```

Dzięki ostatniemu wierszowi, host **moria** jest także znany pod nazwą UUCP. Zauważ, że koszt musi być określony jako `DAILY/2`, ponieważ połączenie dwa razy dziennie w rzeczywistości zmniejsza koszt o połowę.

Dzięki informacjom z takich plików map, *pathalias* jest w stanie obliczyć optymalną trasę do dowolnego ośrodka docelowego umieszczonego w pliku ścieżek i wygenerować na tej podstawie bazę aliasów ścieżek, która z kolei może być wykorzystana do obsługi rutingu do tych ośrodków.

pathalias pełni także kilka innych funkcji, jak ukrywanie ośrodka (tzn. powodowanie, że dostęp jest możliwy tylko przez gateway). Szczegóły oraz pełną listę kosztów łączy znajdziesz na stronie podręcznika elektronicznego *pathalias*.

Komentarze w pliku map przeważnie zawierają dodatkowe informacje o opisywanych ośrodkach. Ta informacja jest podawana według ściśle określonego schematu, i dzięki temu można ją uzyskać z map. Na przykład program o nazwie *uuwho* wykorzystuje bazę danych, stworzoną na podstawie plików map, do wyświetlania elegancko sformatowanych informacji. Gdy zarejestrujesz swój ośrodek w organizacji, która dystrybuuje pliki map swoim członkom, przeważnie musisz wypełnić wpis do

pliku map. Poniżej podajemy przykładowy wpis z pliku map (w rzeczywistości jest to wpis ośrodka Olafa):

```
#N      monad, monad.swb.de, monad.swb.sub.org
#S      AT 486DX50; Linux 0.99
#O      private
#C      Olaf Kirch
#E      okir@monad.swb.de
#P      Kattreinstr. 38, D-64295 Darmstadt, FRG
#L      49 52 03 N / 08 38 40 E
#U      brewhq
#W      okir@monad.swb.de (Olaf Kirch); Sun Jul 25 16:59:32 MET DST 1993
#
monad   brewhq(DAILY/2)
# Domeny
monad = monad.swb.de
monad = monad.swb.sub.org
```

Biały znak występujący po pierwszych dwóch znakach w wierszu to tabulator. Znaczenie większości pól jest oczywiste. Szczegółowy opis dostaniesz z domeny, w której się zarejestrujesz. Najzabawniejsze jest pole `L:`, podaje twoją pozycję geograficzną (szerokość/długość) i jest używane do rysowania map postscriptowych pokazujących wszystkie ośrodki w każdym kraju oraz na świecie*.

Konfigurowanie elma

elm to skrót od słów *electronic mail* (poczta elektroniczna). Jest to jedno z bardziej sensownie nazwanych narzędzi w Uniksie. Zapewnia pełnoekranowy interfejs z dobrze opracowaną funkcją pomocy. Nie będziemy tu tajdawali instrukcji, jak używać *elma*, ale zajmiemy się jego konfiguracją.

Teoretycznie możesz uruchomić *elma* bez konfiguracji i wszystko będzie dobrze działało – jeżeli masz szczęście. Jest jednak kilka opcji, które muszą być ustawione, choć przydają się tylko czasem.

Przy uruchamianiu *elm* odczytuje zestaw zmiennych konfiguracyjnych z pliku *elm.rc* w katalogu */etc/elm*. Następnie próbuje odczytać plik *.elm/elmrc* z twojego katalogu macierzystego. Zwykle sam nie tworzysz tego pliku. Jest on tworzony, gdy wybierzesz w menu opcji *elma* „Save new options".

Zestaw opcji dla prywatnego pliku *elmrc* jest dostępny również w pliku globalnym *elm.rc*. Większość ustawień z twojego pliku prywatnego *elmrc* może zastąpić ustawienia w pliku globalnym.

Opcje globalne elma

W pliku globalnym *elm.rc* musisz ustawić opcje związane z nazwą twojego hosta. Na przykład w browarze wirtualnym plik hosta **vlager** jest następujący:

```
#
# Nazwa hosta
hostname = vlager
```

* Mapy te są regularnie wysyłane do grupy *news.lists.ps-maps*. Ostrzegamy: są OGROMNE.

```
#
# Nazwa domeny
hostdomain = .vbrew.com
#
# Pe□na nazwa domenowa
hostfullname = vlager.vbrew.com
```

Te opcje dają *elmowi* pojęcie o hoście lokalnym. Choć informacje te są rzadko używane, powinieneś je jednak ustawić. Zauważ, że te szczególne opcje działają tylko w pliku globalnym. Jeżeli znajdą się w twoim pliku prywatnym *elmrc*, będą ignorowane.

Narodowe zestawy znaków

Opracowano zestaw standardów i RFC, które wzbogaciły standard RFC-822 o obsługę różnych typów wiadomości, jak czysty tekst, dane binarne, pliki PostScript itp. Te standardy są powszechnie znane jako MIME, czyli uniwersalne rozszerzenie poczty internetowej (*Multipurpose Internet Mail Extensions*). MIME pozwala między innymi, by odbiorca wiedział, czy w czasie pisania wiadomości został użyty inny zestaw znaków, niż standardowe ASCII, czyli na przykład francuskie czy niemieckie znaki diakrytyczne. *elm* w pewnym stopniu te znaki obsługuje.

Do reprezentowania znaków używa się w Linuksie zwykle zestawu ISO-8859-1. Jest on również znany pod nazwą Latin-1. Każda wiadomość wykorzystująca znaki z tego zestawu powinna mieć w nagłówku następującą linię:

```
Content-Type: text/plain; charset=iso-8859-1
```

System odbiorczy powinien rozpoznawać to pole i wyświetlać wiadomość w odpowiedni sposób. Domyślna wartość `charset` dla wiadomości `text/plain` to `us-ascii`.

Aby wyświetlać wiadomości zawierające zestawy znaków inne niż ASCII, *elm* musi wiedzieć, jak te znaki pokazać. Domyślnie, gdy *elm* odbiera wiadomość z polem `charset` o wartości innej niż `us-ascii` (lub typem treści innym niż `text/plain`), próbuje ją wyświetlić za pomocą polecenia *metamail*. Wiadomości wymagające *metamail* są pokazywane z literką M w jednej z pierwszych kolumn na liście wiadomości.

Ponieważ wbudowanym zestawem znaków Linuksa jest ISO-8859-1, wywoływanie *metamail* nie jest konieczne, by wyświetlić wiadomość wykorzystującą ten zestaw znaków. Jeżeli *elm* wie, że urządzenie wyświetlające rozumie standard ISO-8859-1, nie będzie używać *metamail*, ale wyświetli wiadomość bezpośrednio. Można to włączyć, ustawiając poniższą opcję w globalnym pliku *elm.rc*:

```
displaycharset = iso-8859-1
```

Zauważ, że powinieneś ustawić tę opcję nawet wtedy, gdy nie zamierzasz wysyłać ani odbierać wiadomości rzeczywiście zawierających znaki inne niż ASCII. A to dlatego, że ludzie wysyłający takie wiadomości zwykle konfigurują swój program pocztowy tak, by poprawnie wypełniał w nagłówku pole `Content-Type:`, bez względu na to, czy wysyłają wiadomości zapisane w czystym kodzie ASCII, czy nie.

Jednak ustawienie tej opcji w *elmie* nie jest obowiązkowe. Przy wyświetlaniu wiadomości za pomocą wbudowanego programu stronicującego, *elm* wywołuje funkcję biblioteczną wykrywającą, czy każdy ze znaków jest drukowalny. Domyślnie funkcja ta rozpoznaje jedynie znaki ASCII jako drukowalne i wyświetla wszystkie pozostałe jako `^?`. Funkcję tę możesz wyłączyć, ustawiając zmienną środowiskową `LC_CTYPE` na `ISO-8859-1`, która powoduje, że biblioteka uznaje znaki Latin-1 jako drukowalne. Obsługa tej i innych funkcji jest dostępna w Linuksie od wersji 4.5.8 standardowej biblioteki.

Przy wysyłaniu wiadomości zawierającej znaki specjalne z zestawu ISO-8859-1, powinieneś ustawić dwie dodatkowe zmienne w pliku *elm.rc*:

```
charset = iso-8859-1
textencoding = 8bit
```

Powoduje to, że *elm* w nagłówku poczty ustawia zestaw znaków ISO-8859-1 i wysyła wiadomość jako dane 8-bitowe (domyślnie wszystkie znaki są obcinane do 7 bitów).

Oczywiście wszystkie omówione tu opcje związane z zestawem znaków mogą być także ustawiane w prywatnym pliku *elmrc*, tak by indywidualni użytkownicy mogli mieć własne domyślne ustawienia, gdyby globalne im nie odpowiadały.

18



Sendmail

Wprowadzenie do sendmaila

Po wiedzą ci, że nie je steś *prawdziwym* ad mi ni stra to rem Unik sa, je żeli nie edy to wałeś pli ku *sendmail.cf*. Po wiedzą ci również, że je steś sza lo ny, je śli pr óbowałeś to zro bić dwukrotnie.

sendmail jest niezwykle silnym programem pocztowym. Jest on także niezwykle trud ny. Wie le wysiłku kosztuje na ucze nie się go i zro zu mie nie. Ka żdy pro gram, któ- rego opis (książka *Sendmail* au tors twa Bry ana Co stal esa i Erica All mana wy dana przez O'Reilly'ego) zaj muje 1050 stron, jest po strac hem dla wię ks zość lu dzi.

Na szczę ście no we wer sje *sendmaila* są in ne. Nie mu sisz już bez po śred nio edy to wać trud ne go do roz szy fro wa nia pli ku *sendmail.cf*. No wa wer sja za wie ra pro gram kon fi- gu ra cyj ny, któ ry two rzy za cie bie ten plik w opar ciu o du żo prost sze pli ki makr. Nie mu sisz wiec zgłębiać je go złożonej skład ni. Pli ki makr nie wy ma gają te go od cie bie. Wy star czy, że wy pi szesz na zwy funk cji, które chcesz umie ścić w swo jej kon fi gu ra cji i określisz pewne parametry. Tradycyjne narzędzie uniksowe, *m4*, wykorzystuje two je da ne kon fi gu ra cyj ne i – w ce lu stwo rze nia pli ku *sendmail.cf* – łączy je z da ny mi od czy ta ny mi z pli ków wzor co wych za wie rających rze czy wistą skład nię *sendmail.cf*.

W tym roz dziale przed staw imy *sendmail* i opis zemy, jak go za ins talo wać, skon fi gu- rować i przetestować. Za przykład posłuży nam browar wirtualny. Jeżeli dzięki przed staw ionym tu in form acj om uda ci się zmniej szyć oba wy przed kon fi gu ro wa- niem *sendmaila*, być może nabierzesz pew ności sie bie i po dejm iesz sa mod zielnie bar- dziej złożone za dan ia kon fig ura cyj ne.

Instalacja sendmaila

Agent transportowy pocz ty *sendmail* jest dołączany do większości dystrybucji Linuk sa. W ta kim przy pad ku in sta la cja jest sto sun ko wo łatwa. Jed nak z pew nych wzg lędów le piej za in sta lo wać *sendmaila* w po sta ci ko du źr ódłowe go, szc zególnie je- żeli jesteś wyczulony na bezpieczeństwo. Pro gram *sendmail* jest bardzo złożony

i przez lata zdobył wątpliwą reputację programu zawierającego błędy, które ułatwiają włamywanie. Jednym z najlepiej znanych przykładów jest robak internetowy RTM, który zrobił użytek z przekraczania zakresu bufora (ang. *buffer overflow*) – problemu spotykanego w starszych wersjach *sendmaila*. Mówiliśmy o tym krótko w rozdziale 9, *Firewall TCP/IP*. Większość dziur w bezpieczeństwie wykorzystujących przekroczenie zakresu bufora wynika z tego, że wszystkie kopie *sendmaila* na różnych komputerach są identyczne, ponieważ wykorzystywane dane są zapisywane w określonych miejscach. Oczywiście to samo dotyczy *sendmaila* zainstalowanego z dystrybucji Linuksa. Samodzielne kompilowanie *sendmaila* może pomóc zredukować to zagrożenie. Współczesne wersje *sendmaila* są mniej podatne na takie ataki, ponieważ są poddawane niezmiernie dokładnym testom, odkąd, dzięki społeczności Internetu, doceniono bezpieczeństwo.

Kod źródłowy *sendmaila* jest dostępny przez anonimowe FTP pod adresem **ftp.sendmail.org**.

Kompilacja *sendmaila* jest bardzo prosta, ponieważ jego pakiet źródłowy bezpośrednio uwzględnia Linuksa. Kroki wymagane przy kompilacji są następujące:

```
# cd /usr/local/src
# tar xvfs sendmail.8.9.3.tar.gz
# cd src
# ./Build
```

Aby zakończyć instalację uzyskanych plików binarnych, musisz mieć prawa roota:

```
# cd obj.Linux.2.0.36.i586
# make install
```

W tym momencie pliki binarne *sendmaila* są zainstalowane w katalogu */usr/sbin*. W katalogu */usr/bin* powstało kilka dowiązań symbolicznych do plików binarnych *sendmaila*. Powiemy o nich przy okazji omawiania typowych zadań związanych z eksploatacją *sendmaila*.

Przegląd plików konfiguracyjnych

Tradycyjnie *sendmail* był konfigurowany przez systemowy plik konfiguracyjny (zwykle */etc/mail/sendmail.cf* lub w starszych dystrybucjach */etc/sendmail.cf* lub nawet */usr/lib/sendmail.cf*), który nie przypominał żadnego znanego ci dotąd języka. Edycja pliku *sendmail.cf* i dostosowywanie zachowania programu do własnych potrzeb może być przykrym doświadczeniem.

Obecnie *sendmaila* konfiguruje się za pomocą makr o prostej składni. Metoda makr pozwala na generowanie konfiguracji wystarczających dla większości instalacji, ale zawsze masz możliwość poprawienia pliku *sendmail.cf* ręcznie, jeżeli pracujesz w bardziej skomplikowanym środowisku.

Pliki sendmail.cf i sendmail.mc

Program makroprocesora, *m4*, generuje plik *sendmail.cf*, przetwarzając pliki konfiguracyjne makrstworzone przez lokalnego administratora. Dalej ten plik będziemy nazywać *sendmail.mc*.

Proces konfiguracji w zasadzie polega na stworzeniu odpowiedniego pliku *sendmail.mc*, który zawiera makra opisujące żądaną konfigurację. Makra to wyrażenia rozumiane przez makroprocesor *m4* i rozwijane do złożonej składni *sendmail.cf*. Wyrażenia makr składają się z nazwy makra (tekst pisany dużymi literami), która może być połączona z funkcją w języku programowania, i kilku parametrów (tekst w nawiasach), które są używane w trakcie rozwijania makr. Parametry mogą być przekazane dosłownie do pliku *sendmail.cf* lub wykorzystane do zarządzania sposobem przetwarzania makra.

Plik *sendmail.mc* w minimalnej konfiguracji (UUCP lub SMTP z przekazywaniem poczty nielokalnej przez bezpośrednio podłączony inteligentny host) może mieć długość zaledwie 10 czy 15 wierszy, nie licząc komentarzy.

Dwa przykładowe pliki sendmail.mc

Jeżeli jesteś administratorem wielu różnych hostów pocztowych, możesz mieć potrzebę nazwania swoich plików konfiguracyjnych inaczej niż *sendmail.mc*. Zwykle nazywa się je zgodnie z nazwą hosta, czyli w naszym przypadku *vstout.m4*. Nazwa tak naprawdę nie ma znaczenia, ważne, żeby plik wynikowy nazywał się *sendmail.cf*. Nadanie unikalnej nazwy plikowi konfiguracyjnemu każdego hosta pozwala ci przechowywać wszystkie te pliki w jednym katalogu, co jest po prostu wygodne dla administratora. Przyjrzyjmy się dwóm przykładowym plikom konfiguracyjnym, żebyśmy wiedzieli, z czym mamy do czynienia.

Większość konfiguracji *sendmaila* używa obecnie jedynie SMTP. Taka konfiguracja jest bardzo prosta. Przykład 18-1 oczekuje, że do rozwiązywania nazw hostów będzie dostępny serwer DNS, a poza tym przyjmuje i dostarcza całą pocztę dla hostów, używając tylko SMTP.

Przykład 18-1. Przykładowy plik konfiguracyjny vstout.smtp.m4

```
divert(-1)
#
# Przyk□adowy plik konfiguracyjny dla vstout - tylko smtp
#
divert(0)
VERSIONID('@(#)sendmail.mc  8.7 (Linux) 3/5/96')
OSTYPE('linux')
#
# Do□□czenie obs□ugi protoko□ów poczty lokalnej i smtp
MAILER('local')
MAILER('smtp')
#
FEATURE(rbl)
FEATURE(access_db)
# koniec
```

Plik *sendmail.mc* dla **vstout** w browarze wirtualnym został pokazany w przykładzie 18-2. **vstout** używa SMTP do komunikacji ze wszystkimi hostami w sieci LAN browaru. Zauważysz elementy wspólne z wyżej pokazaną konfiguracją wykorzystującą tylko SMTP. Całą pocztę przeznaczoną dla innych hostów **vstout** wysyła przez UUCP do **moria** – swojego hosta przekaźnikowego do Internetu.

Przykład 18-2. Przykładowy plik konfiguracyjny vstout.uucpsmtp.m4

```
divert(-1)
#
# Przyk□adowy plik konfiguracyjny dla vstout
#
divert(0)
VERSIONID('@(#)sendmail.mc  8.7 (Linux) 3/5/96')
OSTYPE('linux')
dnl
# moria jest naszym inteligentnym hostem, wykorzystujemy transport
# "uucp-new".
define('SMART_HOST', 'uucp-new:moria')
dnl
# Obs□uga protoko□ów poczty lokalnej, uucp i smtp.
MAILER('local')
MAILER('smtp')
MAILER('uucp')
LOCAL_NET_CONFIG
# Ta regu□a gwarantuje, □e ca□a poczta lokalna b□dzie
# dostarczana z wykorzystaniem protoko□u SMTP, a wszystko inne
# b□dzie sz□o przez inteligentny host.
R$* < @ $* .$m. > $* $#smtp $@ $2.$m. $: $1 < @ $2.$m. > $3
dnl
#
FEATURE(rbl)
FEATURE(access_db)
# koniec
```

Jeśli porównasz te dwie konfiguracje, możesz dojść do tego, co robi każdy z parametrów. Wyjaśnimy to szczegółowo.

Typowe parametry używane w sendmail.mc

Pewne elementy pliku *sendmail.mc* są obowiązkowe. Innie można pominąć, jeżeli wystarczą ci wartości domyślne. Kolejność definicji w pliku *sendmail.mc* jest następująca:

1. `VERSIONID`
2. `OSTYPE`
3. `DOMAIN`
4. `FEATURE`
5. Lokalne definicje makr
6. `MAILER`
7. Zestawy reguł `LOCAL_*`

W następnych podrozdziałach omówimy każdą z nich po kolei, odwołując się w razie potrzeby do naszych przykładów 18-1 i 18-2.

Komentarze

Wiersze rozpoczynające się w pliku *sendmail.mc* od znaku # nie są analizowane przez *m4* i domyślnie są przepisywane do pliku *sendmail.cf*. Jest to przydatne, jeżeli chcesz skomentować to, co robi twoja konfiguracja w obu plikach – w wejściowym i wyjściowym.

Aby umieścić w pliku *sendmail.mc* komentarze, które *nie* zostaną przeniesione do pliku *sendmail.cf*, możesz użyć dyrektyw *m4*: `divert` i `dnl`. Dzięki `divert(-1)` nic nie będzie wyprowadzane na wyjście, a `divert(0)` umożliwia powrót do stanu domyślnego. Wszystko, co zostanie wygenerowane pomiędzy tymi wierszami, zostanie wyrzucone. W naszym przykładzie użyliśmy tego mechanizmu do stworzenia komentarza, który będzie tylko w pliku *sendmail.mc*. Aby uzyskać ten sam efekt dla pojedynczego wiersza, możesz użyć dyrektywy `dnl`, która dosłownie oznacza „począwszy od następnego wiersza, usuń wszystkie znaki, aż do nowego wiersza włącznie". Jej też użyliśmy w naszym przykładzie.

Są to standardowe funkcje *m4* i więcej na ich temat możesz znaleźć na stronach podręcznika elektronicznego.

VERSIONID i OSTYPE

```
VERSIONID('@(#)sendmail.mc 8.9 (Linux) 01/10/98')
```

Makro `VERSIONID` jest opcjonalne, ale przydatne do zapisywania wersji konfiguracji *sendmaila* w pliku *sendmail.cf*. Często więc będziesz się z nim spotykał. Zalecamy korzystanie z niego. Natomiast musisz pamiętać, by dodać:

```
OSTYPE('linux')
```

Ta definicja należy do najważniejszych. Makro `OSTYPE` powoduje, że są dodawane pliki definicji, które zawierają poprawne wartości domyślne dla twojego systemu operacyjnego. Większość definicji w makro `OSTYPE` ustawia ścieżki do różnych plików konfiguracyjnych, ścieżki i argumenty do programu wysyłającego pocztę oraz lokalizację katalogów, w których *sendmail* przechowuje wiadomości. Standardowy kod źródłowy *sendmaila* zawiera takie pliki dla Linuksa i zostałyby one wciągnięte w poprzednim przykładzie. Niektóre dystrybucje Linuksa, szczególnie Debian, zawierają własne pliki definicji, które są w pełni zgodne ze standardem Linux-FHS. Jeżeli tak jest też w twojej dystrybucji, prawdopodobnie powinieneś użyć tych definicji, zamiast domyślnych definicji dla Linuksa.

Definicja `OSTYPE` powinna być jedną z pierwszych w twoim pliku *sendmail.mc*, gdyż wiele następnych odwołuje się do niej.

DOMAIN

Makro `DOMAIN` przydaje się, gdy chcesz skonfigurować wiele komputerów w tej samej sieci w standardowy sposób. Jeżeli konfigurujesz kilka hostów, prawdopodobnie nie warto go angażować. Zwykle konfigurujesz takie rzeczy, jak nazwa hostów

przekazujących pocztę lub huby, które będą wykorzystywały wszystkie hosty w twojej sieci.

Standardowa instalacja zawiera katalog wzorców makr *m4*, używany do kierowania procesem konfiguracji. Zwykle jest to */usr/share/sendmail.cf* lub coś podobnego. Znajdziesz w nim podkatalog o nazwie *domain* zawierający wzorce specyficzne dla konfiguracji domeny. Aby wykorzystać makro `DOMAIN`, musisz stworzyć swój własny plik makro zawierający standardowe definicje wymagane dla twojego ośrodka i zapisać go w podkatalogu *domain*. Zwykle zapisujesz w nim tylko makrodefinicje unikalne dla twojej domeny, jak inteligentny host czy hosty przekazujące, ale nie jesteś ograniczony tylko do nich.

Kod źródłowy *sendmaila* jest dostarczany wraz z przykładowymi plikami makr domen, na podstawie których możesz stworzyć swoje własne.

Jeżeli zapisałeś swój plik domenowy jako */usr/share/sendmail.cf/domain/vbrew.m4*, w swoim pliku *sendmail.mc* możesz użyć makra `DOMAIN` w następujący sposób:

```
DOMAIN('vbrew')
```

FEATURE

Makro `FEATURE` pozwala dołączyć do konfiguracji *sendmaila* predefiniowane funkcje, które ułatwiają posługiwanie się konfiguracjami. Funkcji tych jest dużo i w tym rozdziale powiemy tylko o kilku najbardziej przydatnych i ważnych spośród nich. Szczegółowy opis dostępnych funkcji możesz znaleźć w pliku *CF*, załączonym do pakietu z kodem źródłowym.

Aby wykorzystać żądaną funkcję, powinieneś w pliku *sendmail.mc* wpisać następujący wiersz:

```
FEATURE(nazwa)
```

gdzie `nazwa` musisz zastąpić nazwą funkcji. Niektóre funkcje przyjmują jeden opcjonalny parametr. Gdybyś chciał użyć czegoś w inny sposób, niż domyślny, powinieneś określić funkcję w następujący sposób:

```
FEATURE(nazwa, parametr)
```

gdzie `parametr` ma znaczenie oczywiste.

Makrodefinicje lokalne

Standardowe pliki konfiguracyjne makr *sendmaila* oferują wiele zmiennych, dzięki którym możesz dostosować konfigurację do swoich potrzeb. Są to tak zwane *makrodefinicje lokalne*. Wiele z nich wymieniono w pliku *CF* w pakiecie z kodem źródłowym *sendmaila*.

Makrodefinicje lokalne są zwykle wywoływane przez podanie nazwy makra oraz argumentu zawierającego wartość, którą chcesz przypisać zmiennej obsługiwanej przez makro. Kilka często używanych makrodefinicji lokalnych omówimy i pokażemy na przykładach w dalszej części tego rozdziału.

Definiowanie protokołów transportowych poczty

Jeśli chcesz, żeby *sendmail* przesyłał pocztę w jakikolwiek inny sposób, niż lokalnie, musisz mu wskazać, jak ma to robić. Ułatwia to makro `MAILER`. Aktualna wersja *sendmaila* obsługuje szereg protokołów transportowych poczty. Niektóre z nich są eksperymentalne, inne są raczej rzadko używane.

W naszej sieci potrzebne są: protokół SMTP do wysyłania i odbierania poczty pomiędzy hostami sieci lokalnej oraz protokół UUCP do wysyłania i odbierania poczty z naszego inteligentnego hosta. Aby to uzyskać, po prostu dołączamy protokoły transportowe `smtp` i `uucp`. Protokół `local` jest dołączany domyślnie, ale można go dla jasności zdefiniować, jeżeli masz na to ochotę. Jeżeli w swojej konfiguracji dołączasz programy pocztowe `smtp` i `uucp`, musisz pamiętać, by `smtp` zawsze umieszczać jako pierwsze.

Poniższa lista opisuje częściej używane protokoły transportowe dostępne dla makra `MAILER`:

`local`
: Ten protokół obejmuje agenta lokalnego używanego do wysyłania poczty do skrzynek pocztowych użytkowników oraz program wysyłający `prog` używany do wysyłania wiadomości do programów lokalnych. Ten protokół jest dołączany domyślnie.

`smtp`
: Ten protokół implementuje prosty protokół przesyłania poczty elektronicznej (SMTP), który jest najczęściej używanym protokołem w Internecie. Gdy go dołączysz, konfigurowane są cztery programy wysyłające pocztę: `smtp` (podstawowe SMTP), `esmtp` (rozszerzone SMTP), `smtp8` (8-bitowe SMTP) i `relay` (stworzony specjalnie do przekazywania poczty pomiędzy hostami).

`uucp`
: Protokół `uucp` daje obsługę dwóch programów wysyłających: `uucp-old`, czyli tradycyjne UUCP, i `uucp-new`, pozwalający na obsłużenie za jednym razem kilku odbiorców.

`usenet`
: Ten program wysyłający pozwala ci na wysyłanie wiadomości bezpośrednio do sieci grup dyskusyjnych Usenetu. Wszelkie wiadomości lokalne skierowane na adres *news.group.usenet* zostaną przekierowane do sieci grup dyskusyjnych na listę *news.group*.

`fax`
: Jeżeli masz zainstalowane oprogramowanie HylaFAX, ten program wysyłający pozwoli ci przekierować na niego pocztę, tak abyś mógł stworzyć gateway pomiędzy pocztą a faksem. W czasie pisania tej książki ta funkcja jest eksperymentalna i więcej informacji na jej temat możesz uzyskać pod adresem *http://www.vix.com/hylafax/*.

Istnieją inne przydatne, ale rzadziej używane protokoły, takie jak `pop`, `procmail`, `mail11`, `phquery` i `cyrus`. Jeżeli obudziliśmy twoją ciekawość, możesz poczytać na ich temat w książce o *sendmailu* lub w dokumentacji dostarczanej w pakiecie kodu źródłowego.

Konfigurowanie rutingu poczty dla hostów lokalnych

Nasz przykład konfiguracji browaru wirtualnego jest zapewne bardziej skomplikowany niż konfiguracja większości rzeczywistych ośrodków. Obecnie najczęściej używa się tylko SMTP i mało kto interesuje się UUCP. W naszej konfiguracji uwzględniliśmy „inteligentny host", który jest używany do obsługi całej poczty wychodzącej. Ponieważ używamy transportu SMTP w naszej sieci lokalnej, musimy poinformować *sendmaila*, żeby nie wysyłał poczty lokalnej przez inteligentnego hosta. Makro `LOCAL_NET_CONFIG` pozwala wstawiać reguły bezpośrednio do pliku wynikowego *sendmail.cf* i w ten sposób modyfikować obsługę poczty lokalnej. W skrócie powiemy więcej na temat reguł podstawiania, ale w tej chwili powinieneś tylko wiedzieć, że dodana w naszym przykładzie reguła mówi, że poczta przeznaczona dla hostów w domenie **vbrew.com** powinna być dostarczana bezpośrednio do hosta adresata za pomocą protokołu SMTP.

Generowanie pliku sendmail.cf

Gdy skończysz edycję pliku konfiguracyjnego *m4*, musisz go przetworzyć, by wygenerować plik */etc/mail/sendmail.cf* odczytywany przez *sendmaila*. Jest to proste, jak widać na poniższym przykładzie:

```
# cd /etc/mail
# m4 /usr/share/sendmail.cf/m4/cf.m4 vstout.uucpsmtp.mc >sendmail.cf
```

To polecenie wywołuje makroprocesor *m4*, któremu dostarcza się nazwy dwóch makrodefinicji do przetworzenia. *m4* przetwarza pliki w podanej kolejności. Pierwszy plik to standardowe makro wzorców *sendmaila* dostarczane w pakiecie kodu źródłowego, a drugi to oczywiście plik zawierający twoje własne makrodefinicje. Wynik polecenia jest przekierowywany do pliku */etc/mail/sendmail.cf*.

Teraz możesz uruchomić *sendmail* z nową konfiguracją.

Interpretacja i pisanie reguł podstawiania

Można pokusić się o stwierdzenie, że najmocniejszą stroną *sendmaila* są reguły podstawiania. Służą *sendmailowi* do określania, jak przetwarzać odebraną wiadomość. *sendmail* przekazuje adresy z *nagłówków* wiadomości do *zestawu reguł* podstawiania (ang. *rulesets*). Reguły podstawiania przetwarzają adres wiadomości z jednej postaci do drugiej i możesz je traktować podobnie jak polecenie twojego edytora, które zastępuje cały tekst pasujący do jakiegoś wzorca innym tekstem.

Każda reguła ma lewą i prawą stronę, oddzielone przynajmniej jednym znakiem tabulatora. Gdy *sendmail* przetwarza pocztę, przegląda reguły podstawiania, szukając

do pasowania po lewej stronie. Jeżeli adres pasuje do lewej strony reguły, jest zastępowany prawą stroną i ponownie przetwarzany.

Polecenia R i S pliku sendmail.cf

W pliku *sendmail.cf* zestawy reguł są definiowane za pomocą poleceń zapisywanych jako `S`*n*, gdzie *n* określa bieżący zestaw reguł.

Same reguły są kodowane jako *R*. Przy odczytywaniu każdego polecenia *R*, reguła jest dodawana do aktualnego zestawu.

Jeżeli używasz tylko pliku *sendmail.mc*, nie musisz w ogóle zawracać sobie głowy poleceniami *S*, gdyż większość makr stworzy je za ciebie. Ręcznie musisz tworzyć tylko reguły *R*.

Zestaw reguł *sendmaila* wygląda tak:

```
Sn
Rlhs rhs
Rlhs2 rhs2
```

Kilka przydatnych makrodefinicji

sendmail wykorzystuje wewnętrznie kilka standardowych makrodefinicji. Najbardziej przydatne z nich przy pisaniu zestawów reguł są:

$j
: Pełna nazwa domenowa danego hosta.

$w
: Nazwa hosta uzyskana na podstawie FQDN.

$m
: Nazwa domeny uzyskana na postawie FQDN.

Te makrodefinicje możemy wykorzystywać w naszych regułach podstawiania. W konfiguracji browaru wirtualnego używane jest makro `$m`.

Lewa strona

Po lewej stronie reguły podstawiania umieszczasz wzorzec, do którego musi pasować adres, który chcesz przekształcić. Większość znaków jest dopasowywana dosłownie, ale jest kilka, które mają szczególne znaczenie. Przedstawiamy je poniżej. Reguły podstawiania dla lewej strony są następujące:

`$@`
: Pasuje dokładnie zero leksemów.

`$*`
: Pasuje zero lub mniej leksemów.

`$+`
: Pasuje jeden lub więcej leksemów.

`$-`
: Pasuje dokładnie jeden leksem.

`$=x`

Pasuje dowolna fraza z klasy *x*.

`$~x`

Pasuje dowolne słowo nienależące do klasy *x*.

Leksem to ciąg znaków ograniczony spacjami. Nie ma ani sposobu na umieszczenie spacji w leksemie, ani takiej potrzeby, gdyż wzorce wyrażenia są dostatecznie elastyczne. Gdy adres zostanie dopasowany do reguły, tekst pasujący do każdego ze wzorców wyrażenia zostanie przypisany specjalnym zmiennym, których będziemy używać po prawej stronie. Jedynym wyjątkiem jest tu `$@`, do którego nie pasuje żaden leksem i dlatego nigdy nie generuje on tekstu, który mógłby być wykorzystany po prawej stronie.

Prawa strona

Gdy adres zostanie dopasowany do reguły podstawiania po lewej stronie, oryginalna treść jest usuwana i zastępowana prawą stroną reguły. Wszystkie leksemy po prawej stronie są dosłownie kopiowane, chyba że zaczynają się od znaku dolara. Tak jak po lewej stronie, po prawej można także używać szeregu metasymboli. Są one opisane na poniższej liście. Reguły podstawiania dla prawej strony są następujące:

`$n`

Ten metasymbol jest zastępowany przez *n*-te wyrażenie z lewej strony.

`$[nazwa$]`

Ten metasymbol rozwija nazwę hosta do postaci kanonicznej. Podana nazwa hosta jest zastępowana przez jej postać kanoniczną.

`$(mapa klucz $@argumenty $:domy□lny $)`

To jest bardziej ogólna postać wyszukiwania. Wynik jest rezultatem poszukiwania `klucza` w mapie o nazwie *map* przy przekazaniu `argumentów`. *Mapą* może być dowolna mapa obsługiwana przez *sendmaila*, jak `virtusertable`, którą opisujemy nieco dalej. Jeżeli wyszukiwanie nie zakończy się sukcesem, zostanie przyjęty wynik `domy□lny`. Jeżeli nie zdefiniowaliśmy wyniku domyślnego i wyszukiwanie się nie powiedzie, to dane wejściowe pozostaną niezmienione, a jako wynik zostanie podany `klucz`.

`$>n`

Ten symbol powoduje, że zostanie przetworzona pozostała część wiersza, a następnie zostanie on przekazany zestawowi reguł *n* do oszacowania. Wynik wywołanego zestawu reguł zostanie zapisany jako wynik tej reguły. Ten mechanizm pozwala na wywoływanie zestawów reguł z reguł.

`$#program_wysy□aj□cy`

Ten metasymbol powoduje, że szacowanie zestawów reguł jest zatrzymywane i określa program wysyłający, który powinien być użyty do przesłania wiadomości w kolejnym kroku jej dostarczania. Ten metasymbol powinien być wywoływany tylko z zestawu reguł 0 lub jednej z jego procedur. Jest to końcowy etap

przetwarzania adresu i powinien być realizowany przez dwa następne metasymbole.

`$@`*`host`*
: Ten metasymbol określa hosta, do którego zostanie przekazana wiadomość. Jeżeli docelowy host jest hostem lokalnym, można go pominąć. *`host`* może mieć postać oddzielonej dwukropkami listy hostów docelowych, do których będą podejmowane kolejne próby dostarczenia poczty.

`$:`*`u□ytkownik`*
: Ten metasymbol określa docelowego użytkownika, dla którego jest przeznaczona poczta.

Pasująca reguła podstawiania jest zwykle testowana dopóty, dopóki któreś dopasowanie się nie powiedzie, a wtedy przetwarzanie przechodzi do następnej reguły. Zachowanie to można zmienić, poprzedzając prawą stronę jednym z dwóch metasymboli podanych poniżej. Symbole sterujące pętlą reguły podstawiania dla prawej strony to:

`$@`
: Ten metasymbol powoduje powrót z zestawu reguł z pozostałą częścią prawej strony jako zwracaną wartością. Nie będą szacowane żadne z pozostałych reguł w zestawie.

`$:`
: Ten metasymbol powoduje natychmiastowe zakończenie reguły, ale pozostała część bieżącego zestawu jest szacowana.

Prosty przykład reguły wzorca

Aby lepiej zrozumieć, jak działa zastępowanie wzorców, rozważmy poniższą lewą stronę reguły:

```
$* < $+ >
```

Do tej reguły pasuje „zero lub więcej leksemów, potem znak <, potem jeden lub więcej leksemów i znak >".

Gdyby tę regułę zastosować do `brewer@vbrew.com` lub `Head Brewer < >`, nie pasowałaby. Pierwszy ciąg nie pasowałby, bo nie zawiera znaku <, a drugi, bo do `$+` pasuje *jeden lub więcej* leksemów, a pomiędzy znakami <> leksemów nie ma. W każdym przypadku, gdy nic nie pasuje do reguły, prawa strona nie jest używana.

Gdyby regułę zastosować do `Head Brewer < brewer@vbrew.com >`, pasowałaby, a zmienna `$1` po prawej stronie zostałaby zastąpiona przez `Head Brewer`, zaś `$2` przez `brewer@vbrew.com`.

Gdyby regułę zastosować do `< brewer@vbrew.com >`, pasowałaby, ponieważ do `$*` pasuje *zero* lub więcej leksemów, a `$1` po prawej stronie zostałoby zastąpione ciągiem pustym.

Składnia zestawu reguł

Każdy zestaw reguł *sendmaila* ma do spełnienia swoje własne zadania w przetwarzaniu poczty. Gdy piszesz reguły, trzeba wiedzieć, czego można się spodziewać po danym zestawie. Przyjrzymy się zestawom reguł, które można modyfikować przez skrypty konfiguracyjne *m4*:

LOCAL_RULE_3
: Zestaw reguł 3 jest odpowiedzialny za konwertowanie adresów w dowolnym formacie na format obsługiwany przez *sendmaila*. Oczekiwany wynikowy format ma znajomą postać `cz□□□-lokalna@host.domena`.

  Zestaw reguł 3 umieszcza nazwę hosta konwertowanego adresu pomiędzy znakami <>, by ułatwić przetwarzanie przez następne zestawy reguł. Zestaw reguł 3 jest stosowany, zanim *sendmail* wykonaja jakiekolwiek inne przetwarzanie adresu e-mail, a więc jeżeli chcesz, by *sendmail* był gatewayem z/do systemu używającego jakiegoś niezwykłego formatu adresu, za pomocą `LOCAL_RULE_3` powinieneś dodać regułę konwertującą adresy do standardowej postaci.

LOCAL_RULE_0 i *LOCAL_NET_CONFIG*
: Zestaw reguł 0 jest stosowany przez *sendmaila* do przetwarzania adresów odbiorcy po zestawie reguł 3. Makro `LOCAL_NET_CONFIG` powoduje, że reguły są wstawiane do *drugiej części* zestawu reguł 0.

  Zestaw reguł 0 realizuje dostarczanie wiadomości do odbiorcy, a więc musi dawać w wyniku trzy elementy określające program wysyłający pocztę, hosta i użytkownika. Reguły będą umieszczone przed definicją inteligentnego hosta, którą możesz wstawić. A więc, jeżeli dodasz reguły, które odpowiednio rozwiązują adresy, wszelkie pasujące do reguły adresy nie będą obsługiwane przez inteligentny host. W ten sposób obsługujemy bezpośrednio *smtp* na przykład dla użytkowników sieci lokalnej w naszym przykładzie.

LOCAL_RULE_1 i *LOCAL_RULE_2*
: Zestaw reguł 1 dotyczy wszystkich adresów nadawców, a zestaw reguł 2 wszystkich adresów odbiorców. Oba zestawy zwykle są puste.

Interpretacja reguły w naszym przykładzie

W naszym przykładzie 18-3 używamy makra `LOCAL_NET_CONFIG` do zadeklarowania lokalnej reguły, która sprawia, że każda poczta w naszej domenie jest dostarczana bezpośrednio przy użyciu programu wysyłającego *smtp*. Teraz, gdy wiemy, jak są zbudowane reguły podstawiania, będziemy mogli zrozumieć, jak działa ta reguła. Przyjrzyjmy się jej ponownie.

Przykład 18-3. Reguła podstawiania z vstout.uucpsmtp.m4

```
LOCAL_NET_CONFIG
# Ta regu□a zapewnia, □e ca□a poczta lokalna b□dzie
# dostarczana za pomoc□ protoko□u SMTP, a wszystko inne
# b□dzie sz□o przez inteligentny host.
R$* < @ $* .$m. > $* $#smtp $@ $2.$m. $: $1 < @ $2.$m. > $3
```

Wiemy, że makro `LOCAL_NET_CONFIG` powoduje, że reguła zostanie wstawiona gdzieś pod koniec zestawu reguł 0, ale przed definicją inteligentnego hosta. Wiemy także, że zestaw reguł 0 jest ostatnim wykonywanym i że powinien dawać w wyniku trzy elementy: program wysyłający, użytkownika i hosta.

Możemy zignorować trzy wiersze komentarza – nie robią one nic przydatnego. Sama reguła to wiersz rozpoczynający się od litery `R`. Wiemy, że `R` jest poleceniem *sendmaila*, które dodaje regułę do aktualnego zestawu reguł, czyli w tym przypadku do zestawu 0. Przyjrzyjmy się kolejno lewej i prawej stronie.

Lewa strona wygląda tak: `$* < @ $* .$m. > $*`.

Zestaw reguł 0 oczekuje znaków < i >, ponieważ dostaje dane z zestawu reguł 3. Zestaw reguł 3 konwertuje adresy do standardowej postaci i ułatwia przetwarzanie. Umieszcza także hosta, do którego jest adresowana poczta, pomiędzy znakami <>.

Do tej reguły pasuje każdy adres wyglądający tak: `'DestUser < @ somehost.our-domain > Some Text'`. To znaczy, że każda wiadomość pasuje do każdego użytkownika na hoście w naszej domenie.

Pamiętasz, że tekst dopasowany przez metasymbole po lewej stronie reguły podstawiania jest przypisywany do makrodefinicji używanych po prawej stronie. W naszym przykładzie, do pierwszego `$*` pasuje cały tekst od początku adresu do znaku <. Cały ten tekst zostanie przypisany do zmiennej `$1`, którą można wykorzystać po prawej stronie. Podobnie drugi `$*` w naszej regule jest przypisywany do zmiennej `$2`, a ostatni do `$3`.

To już wystarcza do zrozumienia lewej strony. Do tej reguły pasuje każda wiadomość dla dowolnego użytkownika na dowolnym hoście w naszej domenie. Nazwa użytkownika jest przypisywana do `$1`, nazwa hosta do `$2`, a dalszy tekst do `$3`. Następnie zmienne te są przetwarzane po prawej stronie.

Przyjrzyjmy się teraz, jaki wynik chcemy uzyskać. Prawa strona naszej przykładowej reguły podstawiania wygląda tak: `$#smtp $@ $2.$m. $: $1 < @ $2.$m. > $3`.

Przy przetwarzaniu prawej strony naszej reguły, każdy z metasymboli jest interpretowany i jest dokonywane odpowiednie podstawienie.

Metasymbol `$#` powoduje, że reguła ta daje określony protokół, w naszym przypadku *smtp*.

Znak `$@` odpowiada hostowi docelowemu. W naszym przykładzie host docelowy jest określony jako `$2.$m.`, co daje pełną nazwę domenową hosta w naszej domenie.

Pełna nazwa jest tworzona na podstawie elementu przypisanego do `$2` po lewej stronie reguły i domeny (`.$m.`).

Metasymbol `$:` określa docelowego użytkownika, którego znów uzyskaliśmy z lewej strony i umieściliśmy w zmiennej `$1`.

Zachowujemy zawartość części ujętej w znaki <> i dalszy tekst, używając danych zebranych z lewej strony reguły.

Ponieważ ta reguła daje nam w rezultacie program wysyłający, wiadomość jest przekazywana temu programowi do dostarczenia. W naszym przykładzie wiadomość zostanie przekazana do docelowego hosta przez protokół SMTP.

Konfigurowanie opcji sendmaila

sendmail posiada szereg opcji pozwalających dostosować sposób realizacji różnych zadań. Jest ich dużo, a więc na poniższej liście pokazujemy tylko kilka częściej używanych.

Aby skonfigurować jakąś z tych opcji, możesz albo zdefiniować ją w pliku konfiguracyjnym *m4*, co jest sposobem preferowanym, albo wstawić ją bezpośrednio do pliku *sendmail.cf*. Na przykład, gdybyś chciał, żeby *sendmail* tworzył nowy proces dla każdej wiadomości, która ma być dostarczona, mógłbyś dodać poniższy wiersz do pliku konfiguracyjnego *m4*:

```
define('confSEPARATE_PROC','true')
```

W pliku *sendmail.cf* utworzony zostałby odpowiedni wpis:

```
O ForkEachJob=true
```

Poniższa lista opisuje powszechnie stosowane opcje pliku wejściowego dla makrogeneratora *m4* (i ich odpowiedniki w pliku wyjściowym *sendmail.cf*):

`confMIN_FREE_BLOCKS (MinFreeBlocks)`
: Czasami zdarza się, że jakiś problem uniemożliwia natychmiastowe doręczenie wiadomości, które są kolejkowane w buforze poczty. Jeżeli twój host pocztowy przetwarza dużo poczty, bufor może urosnąć to takich rozmiarów, że zajmie cały system plików dla niego przeznaczony. Aby temu zapobiec, *sendmail* udostępnia opcję `confMIN_FREE_BLOCKS`, dzięki której można określić minimalną liczbę wolnych bloków dysku twardego, przy jakiej wiadomość zostanie przyjęta. Pozwala ci to mieć pewność, że *sendmail* nigdy nie wypełni całego systemu plików, na którym znajduje się katalog bufora. (Domyślnie: 100).

`confME_TOO (MeToo)`
: Gdy jest rozwijany cel poczty, na przykład alias adresu e-mail, zdarza się, że na liście odbiorców pojawi się nadawca. Ta opcja określa, czy autor wiadomości otrzyma kopię, jeżeli pojawi się na rozwinięciu tej liście odbiorców. Dopuszczalne wartości to „true" i „false". (Domyślnie: false).

`confMAX_DAEMON_CHILDREN (MaxDaemonChildren)`
: Gdy *sendmail* odbiera połączenie SMTP z hosta zdalnego, tworzy nową kopię programu do obsługi przychodzącej wiadomości. W ten sposób możliwe jest przetwarzanie przez *sendmaila* wielu jednocześnie przychodzących połączeń. Choć jest to przydatne, każda nowa kopia *sendmaila* zajmuje pamięć komputera. Jeżeli zostanie odebrana niezwykle duża liczba połączeń ze względu na jakiś błąd lub atak złośliwca, możliwe, że *sendmail* zajmie całą pamięć systemu. Ta opcja pozwala ci ograniczyć maksymalną liczbę demonów potomnych, które mogą zostać utworzone. Gdy liczba ta zostanie osiągnięta, nowe połączenia będą

odrzucane, aż któryś z procesów potomnych zakończy pracę. (Domyślnie: niezdefiniowana).

`confSEPARATE_PROC (ForkEachJob)`
: W czasie przetwarzania kolejki poczty i wysyłania wiadomości, *sendmail* przetwarza po jednej wiadomości. Gdy ta opcja jest włączona, *sendmail* będzie tworzył nową kopię procesu dla każdej dostarczanej wiadomości. Jest to szczególnie przydatne, gdy istnieje kilka wiadomości, które stoją w kolejce ze względu na problem z hostem docelowym. (Domyślnie: false).

`confESMTP_LOGIN_MSG (SmtpGreetingMessage)`
: Gdy jest realizowane połączenie z *sendmailem*, wysyłane są pozdrowienia. Domyślnie wiadomość ta zawiera nazwę hosta, nazwę agenta przesyłającego pocztę, numer wersji *sendmaila*, lokalny numer wersji i aktualną datę. RFC-821 określa, że pierwsze słowo pozdrowień powinno być pełną nazwą domenową, ale pozostała część może być skonfigurowana wedle życzenia. Możesz tu określić makra *sendmaila*. Na skutek użycia – zostaną rozwinięte. Jedyną osobą, która zobaczy tę wiadomość, jest administrator systemu diagnozujący problemy z dostarczaniem poczty lub ciekawscy zainteresowani wykryciem konfiguracji twojej maszyny. Możesz urozmaicić to nudne zadanie, dodając do pozdrowień jakieś dowcipne powiedzenia. Słowo „ESMTP" będzie wstawione przez *sendmaila* pomiędzy pierwsze i drugie słowo, aby zasygnalizować zdalnemu hostowi, że obsługujemy protokół ESMTP. (Domyślnie: `$j Sendmail $v/$Z; $b`).

Użyteczne konfiguracje sendmaila

Istnieje wiele możliwych konfiguracji *sendmaila*. Tu taj pokażemy jedynie kilka ważnych typów konfiguracji, które będą użyteczne w wielu instalacjach *sendmaila*.

Ufamy użytkownikom, że ustawią pole From:

Czasem warto nadpisać pole `From:` w wychodzącej wiadomości. Załóżmy, że masz program generujący wiadomości, oparty na WWW. Zwykle wiadomość wydaje się pochodzić od użytkownika, który jest właścicielem procesu serwera WWW. Możemy określić jakiś inny adres źródłowy, aby wydawało się, że poczta pochodzi od kogoś innego lub spod innego adresu na tej maszynie. *sendmail* pozwala wskazać użytkowników, którym można powierzyć robienie czegoś takiego.

Funkcja `use_ct_file` pozwala na określenie i użycie pliku zawierającego nazwy zaufanych użytkowników. Domyślnie zaufana jest niewielka liczba użytkowników (na przykład `root`). Domyślna nazwa pliku wykorzystywanego przez tę funkcję to */etc/mail/trusted-user* w systemach wykorzystujących katalog konfiguracyjny */etc/mail/*, a */etc/sendmail.ct* – w pozostałych. Nazwę i lokalizację tego pliku możesz określić, nadpisując definicję `confCT_FILE`.

Aby włączyć tę funkcję, dodaj `FEATURE(use_ct_file)` do swojego pliku *sendmail.mc*.

Zarządzanie aliasami pocztowymi

Aliasy pocztowe są silną funkcją, pozwalającą na przekierowywanie poczty do skrzynek pocztowych o alternatywnych nazwach użytkowników lub procesów na hoście docelowym. Na przykład powszechne jest przekierowywanie komentarzy i uwag na temat serwera WWW na konto „webmaster". Często na docelowej maszynie nie istnieje użytkownik „webmaster", a jest to alias innego użytkownika. Innе popularne zastosowanie aliasów pocztowych spotykamy w programach serwerów listdyskusyjnych, w których aliasy kierują pocztę przychodzącą do programu serwera list w celu obsłużenia.

Aliasy są zapisywane w pliku */etc/alias*. Program *sendmail* przegląda ten plik, by stwierdzić, jak obsłużyć przychodzące wiadomości. Jeżeli znajdzie w nim wpis zgodny z adresem w wiadomości, przekierowuje wiadomość we wskazane miejsce.

Aliasy pełnią trzy funkcje:

- Stanowią skrót lub dobrze znaną nazwę pozwalające na adresowanie poczty do jednej lub kilku osób.
- Pozwalają na wywoływanie programu z wiadomością jako jego parametrem wejściowym.
- Pozwalają na przesłanie wiadomości do pliku.

Do zachowania zgodności z RFC, wszystkie systemy potrzebują aliasów dla użytkowników **Postmaster** i **MAILER-DAEMON**.

Gdy definiujesz aliasy wywołujące programy lub piszące do programów, zawsze pilnuj bezpieczeństwa, ponieważ *sendmail* działa przeważnie z prawami roota.

Szczegóły dotyczące aliasów pocztowych możesz znaleźć na stronie podręcznika elektronicznego *aliases(5)*. Przykładowy plik *aliases* jest pokazany poniżej.

Przykład 18-4. Przykładowy plik aliases

```
#
# Poni□sze dwa aliasy musz□ by□ obecne dla zachowania
# zgodno□ci z RFC. Wa□ne jest, by wskazywa□y na 'osob□', która
# czyta regularnie poczt□
#
postmaster:     root                           # wpis wymagany
MAILER-DAEMON:  postmaster                     # wpis wymagany
#
#
# demonstracja ró□nych typów aliasów
#
usenet:        janet                          # alias dla osoby
admin:         joe,janet                       # alias dla kilku osób
newspak-users: :include:/usr/lib/lists/newspak  # odczytywanie odbiorców z
                                               # pliku
changefeed:    |/usr/local/lib/gup             # alias wywo□uj□cy program
complaints:    /var/log/complaints             # alias zapisuj□cy wiadomo□□
                                               # do pliku
```

Zawsze, gdy aktualizujesz plik */etc/aliases*, pamiętaj, by uruchomić polecenie:

```
# /usr/bin/newaliases
```

w celu przebudowania bazy danych używanej wewnętrznie przez *sendmail*. Polecenie */usr/bin/newaliases* jest dowiązaniem symbolicznym do wykonywalnego programu *sendmail* i takie wywołanie działa analogicznie do następującego:

```
# /usr/lib/sendmail -bi
```

Polecenie *newaliases* jest po prostu wygodniejsze.

Używanie inteligentnego hosta

Czasem host napotyka pocztę, której nie jest w stanie doręczyć bezpośrednio do żądanego hosta. Dlatego jeden host w sieci powinien zarządzać przesyłaniem wiadomości do hostów zdalnych, z którymi się trudno połączyć. Jest to wygodniejsze niż danie całkowitej swobody wszystkim hostom, gdyż wtedy każdy host niezależnie podejmowałby nieustanne próby nawiązania takiego połączenia.

Istnieje kilka ważnych powodów, które przemawiają za posiadaniem hosta zarządzającego pocztą. Możesz uprościć zarządzanie, nawet jeśli masz tylko jeden host z pocztą skonfigurowaną w taki sposób, by było wiadomo, jak obsługiwać wszystkie typy protokołów pocztowych, jak UUCP, Usenet itp. Wszystkie pozostałe hosty muszą obsługiwać wtedy tylko jeden protokół, przez który będą wysyłały swoją pocztę do takiego centralnego hosta. Hosty pełniące rolę takiego centralnego rutera pocztowego i przekaźnika poczty są nazywane *hostami inteligentnymi* (ang. *smart hosts*). Jeżeli posiadasz inteligentny host, który przyjmuje od ciebie pocztę, możesz wysłać mu dowolną pocztę, a on obsłuży ruting i przekazanie tej wiadomości do żądanej lokalizacji zdalnej.

Innym dobrym zastosowaniem inteligentnego hosta jest zarządzanie przesyłaniem poczty przez prywatny firewall. Firma może zainstalować sieć wykorzystującą niezarejestrowane adresy IP. Sieć prywatna może być podłączona do Internetu przez firewall. Wysyłanie poczty do i z hostów w sieci prywatnej do świata zewnętrznego za pomocą SMTP nie byłoby możliwe w typowej konfiguracji, ponieważ hosty nie są w stanie przyjąć lub nawiązać bezpośredniego połączenia sieciowego z hostami w Internecie. Firma może zdecydować się na zainstalowanie firewalla, który będzie pełnił funkcję inteligentnego hosta pocztowego. Inteligentny host działający na firewallu jest w stanie zestawiać bezpośrednie połączenia sieciowe między hostami w sieci prywatnej a hostami w Internecie. Inteligentny host przyjmowałby wiadomości zarówno z sieci prywatnej, jak i z Internetu, zachowywał je lokalnie, a następnie obsługiwał wysyłanie bezpośrednio do odpowiedniego hosta.

Inteligentne hosty są zwykle używane wtedy, gdy zawiodą już wszelkie inne metody doręczenia. W przypadku firmy z siecią prywatną najpierw warto spróbować dostarczyć pocztę bezpośrednio, a jeżeli to się nie uda, wysłać ją do inteligentnego hosta. Zmniejszy się ruch do hosta inteligentnego, ponieważ pozostałe hosty mogą wysyłać pocztę bezpośrednio do innych hostów w sieci prywatnej.

sendmail używa prostej metody konfigurowania inteligentnego hosta za pomocą funkcji `SMART_HOST`. Zastosujemy ją przy implementacji konfiguracji browaru wirtualnego. Istotna część naszej konfiguracji definiująca inteligentny host jest następująca:

```
define('SMART_HOST', 'uucp-new:moria')
LOCAL_NET_CONFIG
# Ta regu□a sprawia, □e ca□a poczta lokalna b□dzie
# dostarczana za pomoc□ protoko□u SMTP, a wszystko inne
# b□dzie sz□o przez inteligentny host.
R$* < @ $* .$m. > $* $#smtp $@ $2.$m. $: $1 < @ $2.$m. > $3
```

Makro `SMART_HOST` pozwala ci podać host, przez który powinna być przesyłana cała poczta wychodząca, której nie można dostarczyć bezpośrednio. Można w nim również podać używany przez hosta protokół transportowy.

W naszej konfiguracji używamy protokołu `uucp-new` do połączenia przez UUCP z hostem **moria**. Gdybyśmy chcieli skonfigurować *sendmail*, aby używał inteligentnego hosta opartego na SMTP, zamiast powyższego napisalibyśmy:

```
define('SMART_HOST','mail.isp.net')
```

Nie musimy podawać SMTP jako protokołu transportowego, gdyż jest to protokół domyślny.

Czy wiesz, co robi makro `LOCAL_NET_CONFIG` i reguła podstawiania?

Makro `LOCAL_NET_CONFIG` pozwala ci ręcznie dodawać do twojej konfiguracji reguły podstawiania *sendmaila*, definiujące które poczty powinny pozostać w lokalnym systemie pocztowym. W naszym przykładzie użyliśmy reguły, do której pasują wszystkie adresy, w których host należy do naszej domeny (`.$m.`), i dokonujemy podstawienia, dzięki któremu wiadomości są wysyłane bezpośrednio do programu wysyłającego `SMTP`. W tej sytuacji wszelkie wiadomości dla hosta z naszej domeny są kierowane natychmiast do programu wysyłającego SMTP i przekazywane do danego hosta, i nie przechodzą przez inteligentny host, co jest drogą domyślną.

Zarządzanie niechcianymi i niepotrzebnymi pocztami (spam)

Jeżeli zapisałeś się do pocztowej listy dyskusyjnej, umieściłeś swój adres e-mail w witrynie WWW lub wysłałeś artykuł do grupy Usenet, bardzo prawdopodobne, że zaczniesz dostawać niechciane poczty reklamowe. Obecnie niemało ludzi trudni się wyszukiwaniem ich w sieci adresów pocztowych, dodawaniem ich do list, czy sprzedawaniem firmom reklamującym swoje produkty. Ten rodzaj masowego wysyłania poczty nazywany jest popularnie spammingiem.

Darmowy elektroniczny słownik informatyczny (*Free On-Line Dictionary of Computing*) podaje następującą definicję spamu*:

> 2. (zawężenie znaczenia 1, powyżej) Bezkarne wysyłanie dużej liczby niepożądanych wiadomości e-mail w celu promocji produktu lub usługi. Spam w tym znaczeniu jest odpowiednikiem poczty-śmiecia, wysyłanej do „użytkownika".
>
> Wraz z komercyjnym rozwojem sieci w latach 90., pojawiły się osoby oferujące spamming jako „usługę" firmom, które chcą się reklamować w sieci. Robią to przez wysyłanie wiadomości pod adresy e-mail ze swojej listy, grupy Usenet czy listy pocztowej. Takie praktyki

* Słownik ten można znaleźć w postaci pakietu w dystrybucjach Linuksa lub na jego stronie macierzystej *http://wombat.doc.ic.ac.uk/foldoc/*.

wywołały oburzenie, a nawet agresywne reakcje wielu użytkowników sieci przeciwko uprawiającym spamming.

Na szczęście *sendmail* zawiera mechanizmy pomocne w walce z niechcianą pocztą.

Czarna lista

Czarna lista (ang. *Real-time Blackhole list* – RBL) jest dostępną publicznie usługą, która ma pomóc w ograniczeniu rozsyłania niechcianych reklam. Adresy nadawców spamu i hosty, które udało się rozpoznać, są ujawniane w Internecie w postaci bazy danych, do której można zadawać zapytania. Baza powstaje wysiłkiem ludzi, którzy dostali niechcianą pocztę spod jakiegoś adresu e-mail. Na liście pojawiają się też główne domeny, ze względu na wpadki w zabezpieczeniu się przed przyjmowaniem spamu. Choć niektórzy narzekają na sposób selekcji informacji przez osoby utrzymujące listę, jest ona bardzo popularna, a niezgodności są zwykle szybko wyłapywane. Szczegóły na temat działania usługi można znaleźć na stronie macierzystej jej twórców – projektu Mail Abuse Protection System (MAPS) pod adresem *http://maps.vix.com/rbl/*.

Jeżeli włączysz tę funkcję, *sendmail* będzie sprawdzał adres nadawcy każdej przychodzącej wiadomości i porównywał go z czarną listą, by stwierdzić, czy ma przyjąć wiadomość. Jeżeli twój ośrodek jest duży i ma wielu użytkowników, to dzięki tej funkcji możesz zaoszczędzić wiele miejsca na dysku. Jako parametr przyjmuje ona nazwę serwera, z którego ma korzystać. Domyślnie jest to **rbl.maps.vix.com**.

Aby skonfigurować czarną listę, dodaj poniższe makro do swojego pliku *sendmail.mc*:

```
FEATURE(rbl)
```

Gdybyś chciał podać inny serwer RBL, mógłbyś zapisać deklarację w następujący sposób:

```
FEATURE(rbl,'rbl.host.net')
```

Baza dostępu

Alternatywnym systemem, który oferuje większą elastyczność i kontrolę kosztem ręcznej konfiguracji, jest funkcja `access_db`. Baza dostępu pozwala na skonfigurowanie hostów lub użytkowników, od których przyjmujesz pocztę i na rzecz których pocztę przekazujesz.

Kontrola nad tym, do kogo przekazujesz pocztę, jest ważna, gdyż jest to inna technika powszechnie używana przez hosty spammujące do obejścia opisanej właśnie czarnej listy. Zamiast wysyłać do ciebie pocztę bezpośrednio, spammerzy przesyłają ją przez jakiś inny, niepodejrzany host, który na to pozwala. Przychodzące połączenie SMTP nie pochodzi od hosta spamującego go, a od hosta, przez który jest przekazywane. Aby być pewnym, że twój host nie będzie używany w ten sposób, powinieneś przekazywać pocztę tylko na rzecz znanych hostów. *Sendmail* w wersji 8.9.0 i nowszych ma domyślnie wyłączone przekazywanie, a więc będziesz musiał wykorzystać bazę dostępu, by włączyć przekazywanie dla poszczególnych hostów.

Ogólna zasada jest prosta. Gdy zostanie odebrane nowe przychodzące połączenie SMTP, *sendmail* odczytuje informacje z nagłówka i sprawdza bazę dostępu, by zobaczyć, czy powinien przyjąć wiadomość.

Baza dostępu to zbiór reguł opisujących, co robić, gdy wiadomość zostanie odebrana z określonego hosta. Domyślny plik kontroli dostępu nosi nazwę */etc/mail/access*. Tabela ma prosty format. Każdy wiersz tabeli zawiera regułę dostępu. Lewa strona każdej reguły to wzorzec używany do dopasowania adresu nadawcy przychodzącej poczty. Może to być pełny adres e-mail, nazwa hosta lub adres IP. Po prawej stronie wymienione jest działanie, jakie należy podjąć. Istnieje pięć typów działań, które możesz skonfigurować. Są to:

OK
: Przyjęcie wiadomości.

RELAY
: Przyjęcie wiadomości z tego hosta lub od tego użytkownika, nawet jeżeli nie jest przeznaczona dla naszego hosta. To oznacza przyjęcie wiadomości do przekazania do innych hostów.

REJECT
: Odmówienie przyjęcia z ogólną informacją.

DISCARD
: Odrzucenie wiadomości za pomocą programu wysyłającego `$#discard`.

###dowolny tekst
: Zwrócenie błędu z wykorzystaniem ### jako kodu błędu (który powinien być zgodny z RFC-821) i „dowolnego tekstu" jako treści wiadomości.

Przykładowy plik */etc/mail/access* może wyglądać tak:

```
friends@cybermail.com   REJECT
aol.com                 REJECT
207.46.131.30           REJECT
postmaster@aol.com      OK
linux.org.au            RELAY
```

Ta przykładowa konfiguracja odrzuca wszelkie wiadomości odebrane z adresu *friends@cybermail.com*, od hostów z domeny **aol.com** i od hosta o numerze **207.46.131.30**. Następna reguła przyjmuje pocztę od *postmaster@aol.com*, pomimo że poczta z całej domeny jest odrzucana. Ostatnia reguła pozwala na przekazywanie poczty z dowolnego hosta do domeny **linux.org.au**.

Aby uaktywnić funkcję bazy dostępu, użyj w swoim pliku *sendmail.mc* poniższej deklaracji:

```
FEATURE(access_db)
```

Domyślna definicja tworzy bazę danych, używając polecenia `hash -o /etc/mail/access`, które generuje prostą bazę ze zwykłego pliku tekstowego. Jest to wystarczające w większości przypadków. Istnieją inne opcje, które możesz rozważyć, jeżeli zamierzasz stworzyć dużą bazę dostępu. Szczegóły znajdziesz w książce o *sendmailu* lub innej dokumentacji tego programu.

Wyłączanie otrzymywania poczty przez użytkowników

Jeśli masz użytkowników lub automatyczne procesy, którym wolno wysyłać pocztę, ale nie otrzymywać, czasem warto zablokować przyjmowanie wiadomości dla nich przeznaczonych. Wtedy na dysku nie byłyby zapisywane nigdy nie czytane poczty. Funkcja `blacklist_recipients` w połączeniu z `access_db` pozwala ci wyłączyć odbieranie poczty przez użytkowników lokalnych.

Aby włączyć tę funkcję, dodajesz poniższe wiersze do swojego pliku *sendmail.mc*, o ile ich tam jeszcze nie ma:

```
FEATURE(access_db)
FEATURE(blacklist_recipients)
```

Aby zablokować otrzymywanie poczty przez lokalnego użytkownika, dodaj dotyczące go szczegóły do bazy dostępu. Zwykle używasz wpisu typu ###, który zwraca sensowny komunikat błędu do nadawcy, tak by wiedział, że poczta nie została dostarczona. Ta funkcja dotyczy w równym stopniu wszystkich użytkowników wirtualnych domen pocztowych i musisz w specyfikacji bazy danych dołączyć wirtualną domenę pocztową. Przykładowe wpisy w pliku */etc/mail/access* mogłyby być następujące:

```
daemon           550 Daemon does not accept or read mail.
flacco           550 Mail for this user has been administratively disabled.
grump@dairy.org  550 Mail disabled for this recipient.
```

Konfigurowanie obsługi wirtualnych domen pocztowych

Obsługa wirtualnych domen pocztowych pozwala hostowi na przyjmowanie i dostarczanie poczty na rzecz szeregu różnych domen, tak jakby działało kilka oddzielnych hostów. Funkcja ta, w połaczeniu z obsługą wirtualnych serwrów WWW, jest wykorzystywana zwłaszcza przez dostawców aplikacji internetowych. Jest jednak tak łatwa w konfiguracji, że warto się z tym zapoznać, bo nigdy nie wiadomo, czy nie znajdziesz się w sytuacji, gdy będziesz musiał uruchomić wirtualną listę pocztową dla swojego ulubionego projektu Linuksa. A więc opiszemy tu ten proces.

Przyjmowanie poczty dla innych domen

Gdy *sendmail* odbierze wiadomość e-mail, porównuje host adresata za warty w nagłówku poczty z nazwą hosta lokalnego. Jeżeli pasują, *sendmail* przyjmuje wiadomość do dostarczenia lokalnie. Jeżeli są różne, *sendmail* może przyjąć wiadomość i próbować przekazać ją do celu (szczegóły dotyczące konfiguracji *sendmaila* do przyjmowania poczty w celu jej przekazania znajdziesz we wcześniejszym podrozdziale *Baza dostępu*).

Gdy byś chciał skonfigurować domeny wirtualne, musisz przede wszystkim przekonać *sendmail*, że powinien przyjmować pocztę dla domen, które obsługujemy. Na szczęście dosyć łatwo jest to zrobić.

Funkcja `use_cw_file` pozwala nam określić nazwę pliku, w którym znajdują się nazwy domen, dla których *sendmail* przyjmuje pocztę. Aby skonfigurować tę funkcję, do swojego pliku *sendmail.mc* dodaj następującą deklarację:

```
FEATURE(use_cw_file)
```

Domyślna nazwa pliku to */etc/mail/local-host-names* dla dystrybucji używających katalogu konfiguracyjnego */etc/mail/* lub */etc/sendmail.cw* dla tych, które go nie używają. Alternatywnie możesz określić nazwę i lokalizację tego pliku, nadpisując makro `confCW_FILE`:

```
define('confCW_FILE','/etc/virtualnames')
```

Załóżmy, że używamy domyślnej nazwy pliku. Gdybyśmy chcieli obsługiwać wirtualną pocztę dla domen **bovine.net**, **dairy.org** i **artist.org**, musielibyśmy stworzyć następujący plik */etc/mail/local-host-names*:

```
bovine.net
dairy.org
artist.org
```

Gdy to zrobimy i utworzymy odpowiednie rekordy DNS, gdzie nazwy domen wskazują na nasz host, *sendmail* będzie przyjmował pocztę przeznaczoną dla nich tak, jak by była przeznaczona dla naszej rzeczywistej domeny.

Przekazywanie poczty z wirtualnych domen pocztowych pod inne adresy

Funkcja *sendmaila* `virtusertable` ustawia obsługę tablicy użytkowników wirtualnych, gdzie konfigurujemy wirtualne domeny pocztowe. Tablica użytkowników wirtualnych odwzorowuje przychodzące poczty przeznaczone dla *użytkownik@host* na *innyużytkownik@innyhost*. Możesz to traktować jak zaawansowane aliasy pocztowe, przekierowujące nie tylko użytkownika, ale także domenę.

Aby skonfigurować funkcję `virtusertable`, dodaj do swojego pliku *sendmail.mc* następujący wiersz:

```
FEATURE(virtusertable)
```

Domyślnie plik zawierający reguły translacji nosi nazwę */etc/mail/virusertable*. Możesz ją zmienić, podając odpowiedni argument w makrodefinicji. Szczegóły związane z dostępnymi opcjami znajdziesz w dokumentacji *sendmaila*.

Format tablicy użytkowników wirtualnych jest bardzo prosty. Lewa strona każdego wiersza zawiera wzorzec reprezentujący oryginalny adres, a prawa strona zawiera wzorzec, na jaki tam ten adres zostanie odwzorowany.

Poniższy przykład pokazuje trzy możliwe typy wpisów:

```
samiam@bovine.net   colin
sunny@bovine.net    darkhorse@mystery.net
@dairy.org          mail@jhm.org
@artist.org         $1@red.firefly.com
```

W tym przykładzie obsługujemy domeny wirtualne **bovine.net**, **dairy.org** i **artist.org**.

Pierwszy wpis przekierowuje pocztę przesyłaną do użytkownika domeny wirtualnej **bovine.net** na użytkownika lokalnego komputera. Drugi wpis przekierowuje pocztę użytkownika tej samej domeny wirtualnej na użytkownika innej domeny. Trzeci przykład przekierowuje całą pocztę adresowaną do użytkownika domeny wirtualnej **dairy.org** na pojedynczy adres zdalny. No i ostatni wpis przekierowuje całą pocztę użytkownika z domeny **artist.org** na tego samego użytkownika w innej domenie. Na przykład *julie@artist.org* zostałaby przekierowana na *julie@red.firefly.com*.

Testowaniekonfiguracji

Polecenie *m4* przetwarza pliki makrodefinicji wyłącznie zgodnie z własnymi regułami składniowymi, nic bowiem nie wie o poprawnej składni *sendmaila*. Tak więc, jeżeli coś zrobiłeś źle w pliku makrodefinicji, i tak nie będzie żadnych komunikatów błędów. Z tego powodu ważne jest dokładne przetestowanie twojej konfiguracji. Na szczęście w *sendmailu* robi się to łatwo.

sendmail posiada tryb „testowania adresu" pozwalający na sprawdzenie naszej konfiguracji i zidentyfikowanie wszelkich błędów. W tym trybie działania wywołujemy *sendmail* z wiersza poleceń, a on prosi nas o podanie reguły i adresu docelowego. Następnie przetwarza adres, używając zadanej reguły podstawienia i wyświetla wynik po przejściu każdej reguły. Aby włączyć ten tryb w *sendmailu*, wywołujemy go z argumentem `-bt`.

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
>
```

Domyślnie jest używany plik konfiguracyjny */etc/mail/sendmail.cf*. Możesz podać inny plik konfiguracyjny, używając argumentu `-C`. Aby sprawdzić naszą konfigurację, musimy wybrać adresy do przetwarzania, które powiedzą nam, że nasze wymagania co do obsługi poczty zostały spełnione. Aby to pokazać, przetestujemy naszą bardziej skomplikowaną konfigurację UUCP pokazaną w przykładzie 18-2.

Najpierw sprawdzimy, czy *sendmail* jest w stanie dostarczyć pocztę do użytkowników lokalnych. Spodziewamy się, że wszystkie adresy będą przekształcone tak, by korzystały z programu wysyłające go **local** na naszej maszynie:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac
rewrite: ruleset   3    input: isaac
rewrite: ruleset  96    input: isaac
rewrite: ruleset  96  returns: isaac
rewrite: ruleset   3  returns: isaac
rewrite: ruleset   0    input: isaac
rewrite: ruleset 199    input: isaac
rewrite: ruleset 199  returns: isaac
rewrite: ruleset  98    input: isaac
rewrite: ruleset  98  returns: isaac
rewrite: ruleset 198    input: isaac
rewrite: ruleset 198  returns: $# local $: isaac
rewrite: ruleset   0  returns: $# local $: isaac
```

Ten wynik pokazuje nam, jak *sendmail* przetwarza pocztę adresowaną do **isaac** w naszym systemie. Każdy wiersz przedstawia informacje przekazane do zestawu reguł lub rezultat uzyskany po przejściu przez zestaw reguł. Wskazaliśmy *sendmailowi*, że chcielibyśmy użyć zestawu reguł 0 i 3 do przekształcenia adresu. Zestaw reguł 0 jest wywoływany normalnie, a wywołanie zestawu 3 wymusiliśmy, ponieważ domyśl-

nie nie jest testowany. Ostatni wiersz pokazuje, że wynik zestawu reguł 0 w rzeczywistości przekazuje do programu wysyłającego **local**, pocztę adresowaną do użytkownika **isaac**.

Następnie sprawdzimy pocztę adresowaną na adres SMTP: **isaac@vstout.vbrew.com**. Powinniśmy uzyskać ten sam wynik co w poprzednim przykładzie:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac@vstout.vbrew.com
rewrite: ruleset   3    input: isaac @ vstout . vbrew . com
rewrite: ruleset  96    input: isaac < @ vstout . vbrew . com >
rewrite: ruleset  96  returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset   3  returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset   0    input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 199    input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 199  returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset  98    input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset  98  returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 198    input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 198  returns: $# local $: isaac
rewrite: ruleset   0  returns: $# local $: isaac
```

Znów test zakończył się poprawnie. Dalej sprawdzimy pocztę kierowaną na adres typu UUCP: **vstout!isaac**.

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 vstout!isaac
rewrite: ruleset   3    input: vstout ! isaac
rewrite: ruleset  96    input: isaac < @ vstout . UUCP >
rewrite: ruleset  96  returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset   3  returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset   0    input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 199    input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 199  returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset  98    input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset  98  returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 198    input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 198  returns: $# local $: isaac
rewrite: ruleset   0  returns: $# local $: isaac
```

Ten test również się udał. Testy potwierdzają, że każda poczta przyjęta dla użytkowników lokalnych zostanie poprawnie dostarczona bez względu na format adresu. Gdybyś zdefiniował aliasy dla twojego komputera, na przykład hosty wirtualne, powinieneś powtórzyć testy dla każdej z alternatywnych nazw, pod jaką znany jest host, aby sprawdzić, czy również działają poprawnie.

Następnie sprawdzimy, czy poczta adresowana do innych hostów w domenie **vbrew.com** jest dostarczana bezpośrednio do tego hosta przez program wysyłający SMTP:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
```

```
> 3,0 isaac@vale.vbrew.com
rewrite: ruleset   3    input: isaac @ vale . vbrew . com
rewrite: ruleset  96    input: isaac < @ vale . vbrew . com >
rewrite: ruleset  96  returns: isaac < @ vale . vbrew . com . >
rewrite: ruleset   3  returns: isaac < @ vale . vbrew . com . >
rewrite: ruleset   0    input: isaac < @ vale . vbrew . com . >
rewrite: ruleset 199    input: isaac < @ vale . vbrew . com . >
rewrite: ruleset 199  returns: isaac < @ vale . vbrew . com . >
rewrite: ruleset  98    input: isaac < @ vale . vbrew . com . >
rewrite: ruleset  98  returns: isaac < @ vale . vbrew . com . >
rewrite: ruleset 198    input: isaac < @ vale . vbrew . com . >
rewrite: ruleset 198  returns: $# smtp $@ vale . vbrew . com . /
    $: isaac < @ vale  . vbrew . com . >
rewrite: ruleset   0  returns: $# smtp $@ vale . vbrew . com . /
    $: isaac < @ vale  . vbrew . com . >
```

Widzimy, że ten test przekierował wiadomość do programu wysyłającego SMTP, który przekaże go bezpośrednio do hosta **vale.vbrew.com** i użytkownika **isaac**. Ten test potwierdza, że nasza definicja `LOCAL_NET_CONFIG` działa poprawnie. Warunkiem powodzenia tego testu jest rozwiązanie docelowej nazwy hosta, a więc w pliku */etc/hosts* lub w lokalnym DNS-ie musi znajdować się odpowiedni wpis. Aby zobaczyć, co się stanie, jeżeli rozwiązanie nazwy będzie niemożliwe, podajemy nieznany host:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac@vXXXX.vbrew.com
rewrite: ruleset   3    input: isaac @ vXXXX . vbrew . com
rewrite: ruleset  96    input: isaac < @ vXXXX . vbrew . com >
vXXXX.vbrew.com: Name server timeout
rewrite: ruleset  96  returns: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset   3  returns: isaac < @ vXXXX . vbrew . com >
== Ruleset 3,0 (3) status 75
rewrite: ruleset   0    input: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset 199    input: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset 199  returns: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset  98    input: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset  98  returns: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset 198    input: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset  95    input: < uucp-new : moria > isaac < @ vXXXX . vbrew . com >
rewrite: ruleset  95  returns: $# uucp-new $@ moria $: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset 198  returns: $# uucp-new $@ moria $: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset   0  returns: $# uucp-new $@ moria $: isaac < @ vXXXX . vbrew . com >
```

Wynik jest zupełnie inny. Najpierw zestaw reguł 3 zwraca błąd wskazujący, że nazwa hosta nie może zostać rozwiązana. Następnie podejmowana jest próba obsłużenia tej sytuacji przez przekazanie do innej funkcji naszej konfiguracji: inteligentnego hosta. Zadaniem inteligentnego hosta jest obsłużenie wszelkich poczt, których nie da się dostarczyć w inny sposób. Podana w teście nazwa hosta nie daje się rozwiązać i reguły pokazują, że poczta powinna zostać przekazana do inteligentnego hosta **moria** poprzez program wysyłający **uucp-new**. Nasz inteligentny host może mieć lepsze połączenia i będzie wiedział, co zrobić z tym adresem.

Ostatni z naszych testów pokazuje, że każda poczta adresowana do hosta spoza naszej domeny jest przekazywana do naszego hosta inteligentnego. Powinien on dać wynik podobny do tego z poprzedniego przykładu:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac@linux.org.au
rewrite: ruleset   3    input: isaac @ linux . org . au
rewrite: ruleset  96    input: isaac < @ linux . org . au >
rewrite: ruleset  96  returns: isaac < @ linux . org . au . >
rewrite: ruleset   3  returns: isaac < @ linux . org . au . >
rewrite: ruleset   0    input: isaac < @ linux . org . au . >
rewrite: ruleset 199    input: isaac < @ linux . org . au . >
rewrite: ruleset 199  returns: isaac < @ linux . org . au . >
rewrite: ruleset  98    input: isaac < @ linux . org . au . >
rewrite: ruleset  98  returns: isaac < @ linux . org . au . >
rewrite: ruleset 198    input: isaac < @ linux . org . au . >
rewrite: ruleset  95    input: < uucp-new : moria > isaac < @ linux . org . au . >
rewrite: ruleset  95  returns: $# uucp-new $@ moria $: isaac < @ linux . org . au . >
rewrite: ruleset 198  returns: $# uucp-new $@ moria $: isaac < @ linux . org . au . >
rewrite: ruleset   0  returns: $# uucp-new $@ moria $: isaac < @ linux . org . au . >
```

Wynik tego testu pokazuje, że nazwa hosta została rozwiązana i że został on przekazany do naszego inteligentnego hosta. Dowodzi to, że nasza definicja `LOCAL_NET_CONFIG` działa poprawnie i w obu sytuacjach jest obsługiwana dobrze. Ten test także kończy się sukcesem, a więc możemy szczęśliwie przyjąć, że nasza konfiguracja jest poprawna, i zacząć jej używać.

Eksploatowanie sendmaila

Demona *sendmail* można uruchomić na dwa sposoby. Jeden to uruchamianie go z demona *inetd*. Drugi, częściej używany, to uruchomienie *sendmaila* jako samodzielnego demona. Często zdarza się, że programy wysyłające pocztę wywołują *sendmail* jako polecenie użytkownika przyjmujące go do wysłania lokalnie utworzoną pocztę.

Jeżeli uruchamiasz *sendmail* jako samodzielnego demona, wstaw polecenie do pliku *rc*. Wtedy demon *sendmaila* uruchomi się w czasie startu komputera. Najczęściej używana składnia to:

```
/usr/sbin/sendmail -bd -q10m
```

Argument `-bd` mówi *sendmailowi*, że ma działać jako demon. Program rozgałęzi się i będzie działał w tle. Argument `-q10m` mówi, by *sendmail* sprawdzał kolejkę co dziesięć minut. Możesz podać inny czas sprawdzania kolejki.

Aby uruchomić *sendmail* z demona sieciowego *inetd*, używa się następującego wpisu:

```
smtp stream  tcp nowait nobody  /usr/sbin/sendmail -bs
```

Argument `-bs` mówi *sendmailowi*, by używał protokołu SMTP na stdin/stdout, co jest wymagane przy używaniu z *inetd*.

Polecenie *runq* zwykle jest dowiązaniem symbolicznym do pliku binarnego *sendmail* i jest wygodniejszą postacią wywołania:

```
# sendmail -q
```

Gdy *sendmail* jest wywoływany w ten sposób, przetwarza wszystkie wiadomości oczekujące w kolejce. Przy wywoływaniu *sendmaila* z *inetd*, musisz także stworzyć zadanie *cron*, które co jakiś czas uruchamia polecenie *runq* służące do obsługi bufora poczty.

Odpowiedni wpis w tablicy *cron* powinien przypominać coś takiego:

```
# Uruchamiaj bufor poczty co pi□tna□cie minut
0,15,30,45   *   *   *   *   /usr/bin/runq
```

W większości instalacji *sendmail* przetwarza kolejkę co 15 minut, co pokazano w przykładowym pliku *crontab*. Przetwarzanie kolejki polega na próbie wysłania czekającej w niej wiadomości.

Sztuczki i kruczki

Istnieje wiele rzeczy, które możesz robić, aby efektywnie zarządzać *sendmailem*. W pakiecie *sendmaila* znajduje się szereg narzędzi do zarządzania. Przyjrzyjmy się najważniejszym z nich.

Zarządzanie buforem poczty

Poczta, za nim zostanie wysłana, jest kolejkowana w katalogu */var/spool/mqueue*. Katalog ten jest nazywany buforem poczty. Program *sendmail* pozwala na wyświetlenie listy wszystkich wiadomości znajdujących się w kolejce i ich stanu.

Polecenie */var/bin/mailq* jest dowiązaniem symbolicznym do programu *sendmail* i działa tak samo jak wywołanie:

```
# sendmail -bp
```

Wynik pokazuje ID wiadomości, jej rozmiar, czas umieszczenia w kolejce, nadawcę i komunikat opisujący jej aktualny stan. Poniższy przykład przedstawia wiadomość czekającą w kolejce ze względu na jakiś problem:

```
$ mailq
        Mail Queue (1 request)
--Q-ID-- --Size-- -----Q-Time----- ------------Sender/Recipient------------
RAA00275      124 Wed Dec  9 17:47 root
                  (host map: lookup (tao.linux.org.au): deferred)
                                   tarry@tao.linux.org.au
```

Ta wiadomość znajduje się wciąż w kolejce, ponieważ nie można znaleźć adresu IP docelowego hosta.

Możemy spowodować, że *sendmail* będzie przetwarzał wiadomości znajdujące się w kolejce, wydając polecenie */usr/bin/runq*.

Polecenie nie pokazuje żadnego wyniku. *sendmail* rozpocznie w tle przetwarzanie poczty znajdującej się w kolejce.

Wymuszanie przetworzenia kolejki pocztowej na hoście zdalnym

Jeżeli używasz tymczasowego połączenia komutowanego z Internetem, ale masz *stały* adres IP, a host MX zbiera twoją pocztę w czasie, gdy jesteś rozłączony, przydaci się wymuszanie na hoście MX, by przetwarzał kolejkę pocztową zaraz po zestawieniu twojego połączenia.

W dystrybucji *sendmaila* dołączono mały program w Perlu, który ułatwia zadanie programom, obsługującym tę funkcję. Skrypt *etrn* pozwala osiągnąć mniej więcej to samo na hoście zdalnym, co polecenie *runq* na hoście lokalnym. Jeżeli wywołamy polecenie pokazane w poniższym przykładzie:

```
# etrn vstout.vbrew.com
```

wymusimy na hoście **vstout.vbrew.com** przetworzenie całej poczty przeznaczonej dla naszego komputera, a czekającej w kolejce.

Zwykle polecenie to dodaje się do skryptu *ip-up* PPP, tak by było wykonywane zaraz po zestawieniu połączenia sieciowego.

Analizowanie statystyk poczty

sendmail zbiera dane na temat wielkości ruchu pocztowego i informacje na temat hostów, do których dostarczył pocztę. Istnieją dwa polecenia pozwalające na wyświetlenie tej informacji: *mailstats* i *hoststat*.

mailstats

Polecenie *mailstats* wyświetla statystyki na temat liczby wiadomości przetworzonych przez *sendmail*. Na początku wypisywana jest data rozpoczęcia przyjmowania wiadomości, a po niej tabela, która zawiera po jednym wierszu dla każdego skonfigurowanego programu wysyłającego pocztę i wiersz pokazujący sumę wszystkich wiadomości. Każdy wiersz zawiera osiem elementów:

Pole	*Znaczenie*
`M`	Numer programu wysyłającego (protokołu transportowego).
`msgsfr`	Liczba wiadomości odebranych przez program.
`bytes_from`	Łączna liczba kilobajtów wiadomości odebranych przez program.
`msgsto`	Liczba wiadomości wysłanych przez program.
`bytes_to`	Łączna liczba kilobajtów wysłanych przez program.
`msgsrej`	Liczba nieprzyjętych wiadomości.
`msgsdis`	Liczba odrzuconych wiadomości.
`Mailer`	Nazwa programu wysyłającego.

Przykładowy wynik polecenia *mailstats* pokazano poniżej.

Przykład 18-5. Przykładowy wynik polecenia mailstats

```
# /usr/sbin/mailstats
Statistics from Sun Dec 20 22:47:02 1998
M   msgsfr  bytes_from  msgsto  bytes_to  msgsrej  msgsds  Mailer
0       0          0K      19      515K        0       0  prog
3      33        545K       0        0K        0       0  local
5      88        972K     139     1018K        0       0  esmtp
=============================================================
T     121       1517K     158     1533K        0       0
```

Te dane są zbierane, jeżeli opcja *StatusFile* w pliku *sendmail.cf* jest włączona i istnieje plik stanu. Zwykle musisz dodać w pliku *sendmail.cf* coś takiego:

```
# plik stanu
O StatusFile=/var/log/sendmail.st
```

Aby ponownie uruchomić zbieranie statystyk, musisz stworzyć plik statystyk o zerowej długości:

```
> /var/log/sendmail.st
```

i ponownie uruchomić *sendmail*.

hoststat

Polecenie *hoststat* wyświetla informacje o stanie hostów, do których *sendmail* próbował dostarczyć pocztę. Polecenie *hoststat* jest równoważne z następującym wywołaniem *sendmaila*:

```
sendmail -bh
```

Wynik pokazuje każdego hosta w oddzielnym wierszu i przy każdym z nich zaznacza, od kiedy (godzina) są podejmowane próby dostarczenia, oraz uzyskany wtedy komunikat.

Przykład 18-6 to rezultat, jakiego możesz oczekiwać od polecenia *hoststat*. Zauważ, że większość wyników pokazuje, że dostarczenie się powiodło z wyjątkiem **earthlink.net**. Komunikat o stanie może pomóc określić powód niepowodzenia. W tym przypadku upłynął czas oczekiwania na połączenie dla tego, że host nie działał, albo nie dało się do niego dostać w czasie, gdy były podejmowane takie próby.

Przykład 18-6. Przykładowy wynik polecenia hoststat

```
# hoststat
----------Hostname-------------How long ago ------------Results---------
mail.telstra.com.au               04:05:41 250 Message accepted for
scooter.eye-net.com.au         81+08:32:42 250 OK id=0zTGai-0008S9-0
yarrina.connect.com.a          53+10:46:03 250 LAA09163 Message acce
happy.optus.com.au             55+03:34:40 250 Mail accepted
mail.zip.com.au                   04:05:33 250 RAA23904 Message acce
kwanon.research.canon.com.au   44+04:39:10 250 ok 911542267 qp 21186
linux.org.au                   83+10:04:11 250 IAA31139 Message acce
albert.aapra.org.au               00:00:12 250 VAA21968 Message acce
field.medicine.adelaide.edu.au 53+10:04:11 250 ok 910742814 qp 721
copper.fuller.net              65+12:38:00 250 OAA14470 Message acce
amsat.org                       5+06:49:21 250 UAA07526 Message acce
mail.acm.org                   53+10:46:17 250 TAA25012 Message acce
extmail.bigpond.com            11+04:06:20 250 ok
earthlink.net                  45+05:41:09 Deferred: Connection time
```

Polecenie *purgestat* czyści zebrane dane i jest równoważne z następującym wywołaniem *sendmaila*:

```
# sendmail -bH
```

Statystyki będą zbierane, aż ich nie wyczyścisz. Możesz co jakiś czas uruchamiać polecenie *purgestat*, aby ułatwić sobie wyszukiwanie ostatnich wpisów, szczególnie jeżeli twój ośrodek jest obciążony. Możesz także umieścić to polecenie w tablicy *crontab*, tak aby było uruchamiane automatycznie, lub możesz uruchamiać je co jakiś czas ręcznie.

19

Exim



Ten rozdział zwięźle wprowadza w konfigurowanie Exima i omawia jego funkcje. Choć Exim zachowuje się podobnie jak *sendmail*, jego pliki konfiguracyjne są zupełnie inne.

Główny plik konfiguracyjny w większości dystrybucji Linuksa nazywa się */etc/exim.conf* lub */etc/exim/config*, a w starszych konfiguracjach */usr/lib/exim/config*. Plik ten możesz znaleźć, uruchamiając poniższe polecenie:

```
$ exim -bP configure_file
```

Może zajść potrzeba edycji pliku konfiguracyjnego, aby dopasować go do wartości specyficznych dla twojego ośrodka. Przy standardowym konfigurowaniu nie trzeba wiele zmieniać, a działająca konfiguracja rzadko musi być modyfikowana.

Domyślnie Exim natychmiast przetwarza i rozsyła wszystkie przychodzące wiadomości. Jeżeli masz stosunkowo duży ruch, możesz skonfigurować Exima tak, by zbierał wiadomości w tak zwanej *kolejce* i przetwarzał je łącznie jedynie co jakiś czas.

Przy obsłudze poczty w sieci TCP/IP, Exim często działa w trybie demona: w czasie uruchamiania systemu jest wywoływany z */etc/init.d/exim** i przechodzi w tło, gdzie czeka na przychodzące połączenia TCP na porcie SMTP (zwykle port 25). Jest to korzystne, gdy spodziewasz się dużego ruchu, gdyż Exim nie musi uruchamiać się dla każdego przychodzącego połączenia. Alternatywnie, *inetd* może zarządzać portem SMTP i Exima, gdy nadejdzie połączenie na ten port. Taka konfiguracja może się przydać, gdy masz ograniczoną wielkość pamięci i niewielki ruch.

Exim ma skomplikowany zestaw opcji wiersza poleceń, a wiele z nich przypomina te z *sendmaila*. Zamiast samemu trudzić się nad dopasowaniem opcji do swoich potrzeb, możesz zaimplementować najpopularniejsze typy operacji, wywołując klasyczne polecenia, jak *rmail* czy *rsmtp*. Są to dowiązania symboliczne do Exima (a jeśli

* Inne możliwe lokalizacje to */etc/rc.d/init.d* i *rc.inet2*. Ta ostatnia jest często spotykana w systemach korzystających ze struktury plików w katalogu */etc* typowej dla BSD.

ich nie ma, możesz je łatwo utworzyć). Gdy uruchomisz jedno z tych poleceń, Exim sprawdzi użytą przez ciebie nazwę i ustawi sam odpowiednie opcje.

Istnieją dwa dowiązania do Exima, które powinieneś mieć bez względu na wszystko: */usr/bin/rmail* i */usr/sbin/sendmail**. Gdy piszesz wiadomość i wysyłasz ją za pomocą agenta, na przykład *elm*, jest ona przekazywana do *sendmaila* lub *rmaila* w celu dostarczenia i dlatego zarówno */usr/sbin/sendmail*, jak i */usr/bin/rmail* powinny wskazywać na Exima. Lista adresatów wiadomości jest przekazywana do Exima w wierszu poleceń** To samo dzieje się z pocztą przychodzącą przez UUCP. Wpisując poniższe wiersze, możesz skonfigurować żądane ścieżki tak, by wskazywały na Exima:

```
$ ln -s /usr/sbin/exim /usr/bin/rmail
$ ln -s /usr/sbin/exim /usr/sbin/sendmail
```

Gdy byś chciał się zagłębić w dalsze szczegóły konfiguracji Exima, powinieneś przeczytać jego pełną specyfikację. Jeżeli nie ma jej w twojej ulubionej dystrybucji Linuksa, możesz ją znaleźć w źródłach Exima lub przeczytać w wersji elektronicznej na witrynie Exima pod adresem *http://www.exim.org*.

Eksploatowanie Exima

Przed uruchomieniem Exima musisz się zdecydować, czy chcesz, żeby obsługiwał on przychodzącą pocztę SMTP jako samodzielny demon, czy jako program zarządzany przez *inetd*, który kontroluje port SMTP i wywołuje Exima tylko wtedy, gdy klient żąda połączenia SMTP. Zwykle na serwerach pocztowych lepiej sprawdza się demon, ponieważ dużo mniej obciąża maszynę niż Exim uruchamiany oddzielnie dla każdego połączenia. Ponieważ serwer pocztowy dostarcza większość przychodzącej poczty bezpośrednio do adresatów, powinieneś na pozostałych hostach wybrać działanie przez *inetd*.

Bez względu na to, który tryb pracy wybierzesz, musisz mieć w swoim pliku */etc/services* następujący wpis:

```
smtp      25/tcp   # Simple Mail Transfer Protocol
```

Definiuje on numer portu TCP, który jest używany do połączeń SMTP. Numer portu 25 jest standardowo zdefiniowany przez RFC-1700 (*Assigned Numbers*).

Gdy uruchomisz Exima w trybie demona, przechodzi on do przetwarzania w tle i czeka na połączenie na porcie SMTP. Gdy połączenie nadejdzie, rozgałęzia się i jego proces potomny prowadzi konwersację SMTP z procesem hosta po drugiej stronie. Demon Exim zwykle jest uruchamiany przez wywołanie ze skryptu *rc* w czasie startu komputera. Służy do tego następujące polecenie:

```
/usr/sbin/exim -bd -q15m
```

* Jest to nowa standardowa lokalizacja *sendmaila* zgodna ze standardem systemu plików Linuksa. Innym, często spotykanym miejscem jest */usr/lib/sendmail*, które może być używane przez programy pocztowe, które nie są specjalnie konfigurowane dla Linuksa. Obie nazwy możesz zdefiniować jako dowiązania symboliczne do Exima, aby programy i skrypty wywołujące *sendmail* tak naprawdę uruchamiały i używały do swoich celów Exima.

** Niektóre agenty używają jednak protokołu SMTP, by przekazać wiadomości do agenta transportowego. Wywołują go wtedy z opcją *–bs*.

Opcja *–bd* włącza tryb demona, a *–q15m* powoduje, że wiadomości zebrane w kolejce są obsługiwane co 15 minut.

Gdybyś chciał użyć *inetd*, twój plik */etc/inetd.conf* powinien zawierać następujący wiersz:

```
smtp      stream   tcp nowait root /usr/sbin/exim in.exim -bs
```

Pamiętaj, że musisz spowodować ponowne przeczytanie pliku *inetd.conf* przez proces *inetd*, wysyłając do niego sygnał `HUP` po dokonaniu niezbędnych zmian*.

Tryb demona i *inetd* wykluczają się wzajemnie. Jeżeli uruchomisz Exima jako demona, powinieneś zakomentować wiersz usługi `smtp` w pliku *inetd.conf*. I odwrotnie, gdy uruchamiasz Exima przez *inetd*, upewnij się, że nie masz skryptu *rc* uruchamiającego go w trybie demona.

Wykonując połączenie przez telnet na port SMTP swojej maszyny, możesz sprawdzić, czy Exim jest poprawnie skonfigurowany do odbierania wiadomości SMTP. Oto jak powinno wyglądać poprawne połączenie:

```
$ telnet localhost smtp
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 richard.vbrew.com ESMTP Exim 3.13 #1 Sun, 30 Jan 2000 16:23:55 +0600
quit
221 richard.vbrew.com closing connection
Connection closed by foreign host.
```

Jeżeli ten test nie spowoduje pokazania banera SMTP (wiersza rozpoczynającego się kodem 220), sprawdź, czy proces demona Exim istnieje lub czy *inetd* jest poprawnie skonfigurowany. Jeżeli to nie rozwiąże problemu, a w pliku konfiguracyjnym nie ma błędów, zajrzyj do plików log Exima (opisanych dalej).

Jeżeli twoja poczta nie dochodzi

Istnieje szereg funkcji pomagających rozwiązywać problemy z instalacją. Pierwszym miejscem, jakie należy sprawdzić, są pliki log Exima. W systemach linuksowych normalnie znajdują się one w katalogu */var/log/exim/log* i nazywają się *exim_mainlog*, *exim_rejectlog* i *exim_paniclog*. W innych systemach operacyjnych często są umieszczane w katalogu */var/spool/exim/log*. Jeśli jeszcze nie wiesz, gdzie się znajdują pliki log Exima w twoim systemie, uruchom poniższe polecenie:

```
exim -bP log_file_path
```

Główny plik log opisuje wszystkie transakcje, plik log reject zawiera szczegóły dotyczące wiadomości, które zostały odrzucone ze względu na przyjętą politykę, a plik log panic zawiera wiadomości związane z błędami konfiguracyjnymi i tym podobnymi.

* Użyj polecenia `kill-HUP pid`, gdzie *pid* oznacza ID procesu *inetd* uzyskane na podstawie wyniku polecenia *ps*.

Poniżej pokazano typowe wpisy w głównym pliku log. Każdy wpis to jeden wiersz tekstu, rozpoczynający się od daty i czasu. Tutaj zostały one podzielone na kilka wierszy, by zmieściły się na stronie:

```
2000-01-30 15:46:37 12EwYe-0004WO-00 <= jack@vstout.vbrew.com
  H=vstout.vbrew.com [192.168.131.111] U=exim P=esmtp S=32100
  id=38690D72.286F@vstout.vbrew.com
2000-01-30 15:46:37 12EwYe-0004WO-00 => jill@vbrew.com>
  D=localuser T=local_delivery
2000-01-30 15:46:37 12EwYe-0004WO-00 Completed
```

Te wpisy pokazują, że wiadomość od *jack@vstout.vbrew.com* do *jill@vbrew.com* została poprawnie dostarczona do skrzynki pocztowej na hoście lokalnym. Przyjęcie wiadomości oznacza się symbolem <=, a nadania – symbolem =>.

Istnieją dwa rodzaje błędów dostarczenia: stały i tymczasowy. Błąd stały uwidacznia się w pliku log w pokazany poniżej sposób i jest oznaczony dwoma gwiazdkami (**):

```
2000-01-30 14:48:28 12EvcH-0003rC-00 ** bill@lager.vbrew.com
  R=lookuphost T=smtp: SMTP error from remote mailer after RCPT TO:
  <bill@lager.vbrew.com>: host lager.vbrew.com [192.168.157.2]:
  550 <bill@lager.vbrew.com>... User unknown
```

Jeśli taki błąd wystąpi, Exim wysyła do nadawcy raport z błędnego dostarczenia, często nazywany *wiadomością odbitą* (ang. *bounced message*).

Błędy tymczasowe są oznaczane symbolem ==:

```
2000-01-30 12:50:50 12E9Un-0004Wq-00 == jim@bitter.vbrew.com
  T=smtp defer (145): Connection timed out
```

Ten błąd jest typowy dla sytuacji, w której Exim prawdopodobnie rozpoznał, że wiadomość powinna zostać dostarczona do hosta zdalnego, ale nie jest w stanie połączyć się z usługą SMTP na tym hoście. Na przykład host jest wyłączony lub przytrafił się jakiś problem z siecią. Gdy wiadomość zostanie *odrzucona* (ang. *deferred*) w ten sposób, pozostaje w kolejce Exima i co jakiś czas jest podejmowana próba jej ponownego wysłania. Jednak jeżeli w określonym czasie (zwykle kilka dni), żadna próba się nie powiedzie, pojawi się błąd stały i zostanie wysłana wiadomość odbita.

Jeżeli na podstawie komunikatu błędu generowanego przez Exima nie jesteś w stanie zlokalizować problemu, możesz włączyć komunikaty debugujące. Robi się to przez opcję *–d*, po której opcjonalnie można podać żądany poziom dokładności wyświetlanych informacji (maksymalnie 9). Exim wyświetla raport na ekranie. Być może z niego dowiesz się, gdzie tkwi błąd.

Kompilowanie Exima

Exim jest wciąż w stadium intensywnego rozwoju. Wersja załączona w dystrybucji Linuksa nigdy nie jest tą najnowszą. Jeżeli potrzebujesz funkcji lub poprawki, która istnieje w nowszej wersji, musisz zdobyć kod źródłowy i skompilować go samodzielnie. Najnowszą wersję można znaleźć na stronie WWW Exima pod adresem *http://www.exim.org*.

Linux jest jednym z wielu systemów operacyjnych, dla którego istnieje konfiguracja w kodzie źródłowym Exima. Aby skompilować go w Linuksie, powinieneś dokonać edycji pliku *src/EDITME* i umieścić wynik w pliku o nazwie *Local/Makefile*. W pliku *src/EDITME* znajdują się komentarze, które informują, do czego służą poszczególne ustawienia. Na koniec uruchom *make*. Szczegółowe informacje na temat kompilacji Exima znajdziesz w jego podręczniku obsługi.

Tryby dostarczania poczty

Jak wspomnieliśmy, Exim może bezzwłocznie dostarczać wiadomości lub kolejkować je do późniejszego przetwarzania. Wszystkie przychodzące wiadomości są zachowywane w podkatalogu *input* katalogu */var/spool/exim*. Gdy kolejkowanie nie działa, proces dostarczania jest uruchamiany po nadejściu każdej wiadomości. W przeciwnym razie wiadomość jest pozostawiana w kolejce, aż proces *queuerunner* ją pobierze. Kolejkowanie może być bezwarunkowe, jeżeli ustawimy w pliku konfiguracyjnym *queue_only*, lub realizowane warunkowo przy średnim obciążeniu systemu w czasie jednej minuty, jeżeli ustawimy:

```
queue_only_load = 4
```

W tym wypadku wiadomości są kolejkowane, jeżeli obciążenie systemu przekroczy 4*.

Jeżeli twój host nie jest na stałe połączony z Internetem, możesz zechcieć włączyć kolejkowanie dla adresów zdalnych, pozwalając Eximowi na natychmiastowe dostarczanie poczty lokalnej. Możesz to zrobić, ustawiając w pliku konfiguracyjnym:

```
queue_remote_domains = *
```

Jeżeli włączysz dowolne kolejkowanie, musisz pamiętać o regularnym sprawdzaniu kolejek, najlepiej co 10 lub 15 minut. Nawet jeżeli opcje kolejkowania nie są jawnie włączone, trzeba sprawdzać kolejki pod kątem wiadomości odrzuconych ze względu na tymczasowe błędy w dostarczaniu. Jeżeli uruchomisz Exima w trybie demona, musisz dodać w wierszu poleceń opcję *–q15m* przetwarzającą kolejkę co 15 minut. Możesz także wywołać *exim –q* z *crona* co zadany okres czasu.

Aktualną kolejkę możesz obejrzeć, wywołując Exima z opcją *–bp*. To samo możesz uzyskać, tworząc dowiązanie *mailq* do Exima i wywołując *mailq*:

```
$ mailq
2h    52K 12EwGE-0005jD-00 <sam@vbrew.com>
      D bob@vbrew.com
          harry@example.net
```

Widzimy, że w kolejce czeka jedna wiadomość od *sam@vbrew.com* adresowana do dwóch osób. Została ona poprawnie dostarczona do *bob@vbrew.com*, ale jeszcze nie dotarła do *harry@example.net*, choć czeka w kolejce od dwóch godzin. Rozmiar wiadomości to 52 KB, a ID za pomocą którego Exim ją identyfikuje to `12EwGE-0005jD-00`. Zaglądając do indywidualnego pliku log wiadomości, *msglog*, który znajduje się w kata-

* Obciążenie systemu jest standardową uniksową miarą średniej liczby procesów, które są kolejkowane i oczekują na wykonanie. Polecenie *uptime* pokazuje średnie obciążenia za następujące okresy czasu: minutę, 5 i 15 minut.

logu buforowym Exima, możesz stwierdzić, dla czego wiadomość nie została jeszcze dostarczona. Łatwo to zrobić, używając opcji *–Mvl*:

```
$ exim -Mvl 12EwGE-0005jD-00
2000-01-30 17:28:13 example.net [192.168.8.2]: Connection timed out
2000-01-30 17:28:13 harry@example.net: remote_smtp transport deferred:
  Connection timed out
```

Indywidualne pliki log zawierają kopię wpisów log dla każdej wiadomości, a więc możesz je łatwo przeglądać. Tę samą informację możesz uzyskać z głównego pliku log, używając narzędzia *exigrep*:

```
$ exigrep 12EwGE-0005jD-00 /var/log/exim/exim_mainlog
```

Potrwa to nieco dłużej, szczególnie w obciążonym systemie, gdzie pliki log są duże. Narzędzie *exigrep* przydaje się przy poszukiwaniu informacji o większej liczbie wiadomości. Jego pierwszym argumentem jest wyrażenie regularne i pokazuje wszystkie wiersze związane z wiadomościami, które mają co najmniej jeden wiersz pasujący do wyrażenia. Tym sposobem można go używać do wybrania tych wszystkich wiadomości, które są adresowane na jeden zadany adres, lub wszystkich tych, które są przeznaczone dla zadanego hosta lub stamtąd pochodzą.

Jeśli chcesz zobaczyć sobie ogólnie, co robi Exim, uruchom polecenie *tail* z głównym plikiem log. Możesz też uruchomić narzędzie *eximon* dostarczane wraz z Eximem. Jest to aplikacja X11, która daje przesuwający się obraz głównego logu i pokazuje listę wiadomości, które czekają na dostarczenie, oraz pewne statystyki aktywności dostarczania.

Różne opcje konfiguracyjne

Oto kilka innych przydatnych opcji, które możesz ustawiać w pliku konfiguracyjnym.

message_size_limit
: Ustawienie tej opcji ogranicza rozmiar wiadomości przyjmowanej przez Exima.

return_size_limit
: Ustawienie tej opcji ogranicza liczbę przychodzących wiadomości, które Exim będzie zwracał w ramach wiadomości odbitej.

deliver_load_max
: Jeżeli obciążenie systemu osiągnie zadaną tą opcją wartość, dostarczanie wszelkich wiadomości zostanie zawieszone, choć wciąż będą one przyjmowane.

smtp_accept_max
: Jest to maksymalna liczba jednocześnie przychodzących połączeń SMTP, które Exim może przyjąć.

log_level
: Ta opcja kontroluje liczbę danych zapisywanych do pliku log. Istnieją pewne opcje o nazwach rozpoczynających się od *log_*, które kontrolują zapisywanie określonych informacji.

Ruting i dostarczanie poczty

Exim dzieli dostarczanie poczty na trzy różne zadania: ruting, zarządzanie i przesyłanie. Istnieje kilka modułów kodu dla każdego typu i każdy konfiguruje się oddzielnie. Zwykle w pliku konfiguracyjnym dostosowuje się kilka różnych ruterów, modułów zarządzających i przesyłających.

Rutery rozwiązują adresy zdalne, aby było wiadomo, do którego hosta powinna być wysłana wiadomość i którego protokołu transportowego należy użyć. W przypadku hostów podłączonych do Internetu, zwykle istnieje jeden ruter, który realizuje rozwiązywanie przez przeszukiwanie domeny w DNS-ie. Ewentualnie może być jeden ruter, który obsługuje adresy hostów w sieci lokalnej, i drugi, który wysyła pozostałe wiadomości do *inteligentnego hosta*, na przykład serwera pocztowego dostawcy Internetu.

Adresy lokalne są przekazywane do programu zarządzającego. Takich programów jest zwykle kilka. Obsługują one aliasy i przekazywanie oraz identyfikują skrzynki lokalne. Listy pocztowe mogą być obsługiwane przez programy zarządzające aliasami i przekazywaniem. Jeżeli adres posiada alias lub został przekierowany, nowo utworzone adresy są obsługiwane niezależnie przez rutery lub programy zarządzające, o ile jest taka potrzeba. Najczęstszym przypadkiem będzie dostarczanie do skrzynki pocztowej, ale wiadomości mogą być także przekazane przez potok do polecenia lub doklejone do pliku innego niż domyślna skrzynka pocztowa.

Moduł transportowy jest odpowiedzialny za implementację metod dostarczania, na przykład za wysłanie wiadomości przez łącze SMTP lub umieszczenie jej w określonej skrzynce pocztowej. Rutery i programy zarządzające decydują, którego modułu transportowego użyć dla danego adresata. Jeżeli moduł transportowy nie zadziała, Exim generuje wiadomość od bitą lub chwilowo odkłada adres, aby później ponowić próbę.

W Eximie masz pełną swobodę konfigurowania tych zadań. Dla każdego z nich dostępne jest kilka sterowników, z których możesz wybrać potrzebny. Opisujesz je w różnych sekcjach pliku konfiguracyjnego Exima. Najpierw definiowane są protokoły transportowe, po nich moduły zarządzające, a na końcu rutery. Nie ma wbudowanych wartości domyślnych, choć Exim jest rozpowszechniany z domyślnym plikiem konfiguracyjnym, który uwzględnia proste przypadki. Gdybyś chciał zmienić politykę rutowania Exima lub zmodyfikować protokół transportowy, łatwiej jest rozpocząć od domyślnej konfiguracji i dokonywać w niej zmian, niż próbować stworzyć plik konfiguracyjny od zera.

Ruting wiadomości

Gdy Exim dostanie adres, na który ma dostarczyć pocztę, najpierw sprawdza, czy domena jest obsługiwana przez host lokalny, porównując ją z listą zawartą w zmiennej konfiguracyjnej `local_domains`. Jeżeli ta opcja nie jest ustawiona, nazwa hosta lokalnego jest używana tylko w domenie lokalnej. Jeżeli jesteśmy w domenie lokal-

nej, adres jest przekazywany do modułów zarządzających. W przeciwnym razie jest przekazywany do ruterów, aby stwierdziły, gdzie przesłać wiadomość*.

Dostarczanie wiadomości na adresy lokalne

Adres lokalny to przeważnie nazwa użytkownika. Jeśli ma taką postać, wiadomość jest dostarczana bezpośrednio do skrzynki pocztowej użytkownika */var/spool/mail/* `nazwa-u□ytkownika`. Do innych przypadków zaliczamy aliasy, nazwy list pocztowych i przekazywanie poczty przez użytkownika. Wtedy adres lokalny jest rozwijany do listy adresów, które mogą być lokalne lub zdalne.

Poza takimi „normalnymi” adresami, Exim może obsługiwać inne typy celów wiadomości lokalnych o innym miejscu przeznaczenia, takim jak nazwy plików i potoki poleceń. Jeśli chodzi o dostarczanie do pliku, Exim dokleja wiadomość, a jeżeli jest taka potrzeba, tworzy nowy plik. Cele w postaci pliku i potoku nie są typowymi adresami, a więc nie możesz wysłać poczty, powiedzmy, pod */etc/passwd@vbrew.com* i oczekiwać, że plik *passwd* zostanie nadpisany. Dostarczanie pod adresy specjalne jest możliwe tylko, jeżeli istnieje jawnie przekierowanie lub pliki aliasów. Zwróć jednak uwagę, że */etc/passwd@vbrew.com* jest składniowo poprawnym adresem e-mail, ale jeżeli Exim odbierze adresowaną na niego wiadomość, zwykle będzie szukał użytkownika o nazwie */etc/passwd*, co zakończy się fiaskiem i wiadomość zostanie odbita.

Na liście aliasów lub w pliku przekierowania *nazwapliku* zaczyna się od ukośnika (/) i ma postać, która nie spełnia warunków składni pełnego domenowego adresu e-mail. Na przykład */tmp/junk* w pliku przekierowania lub w pliku aliasów jest interpretowane jako nazwa pliku, ale */tmp/junk@vbrew.com* jako adres e-mail, choć prawdopodobnie niezbyt przydatny. Jednak adresy tego typu można spotkać przy wysyłaniu poczty przez gatewaye X.400, ponieważ adresy X.400 rozpoczynają się od ukośnika.

Podobnie *polecenie w potoku* może być dowolnym poleceniem Uniksa poprzedzonym znakiem potoku (|), o ile ciąg nie może być uznany za poprawny, domenowy adres e-mail. Jeżeli nie zmienisz konfiguracji, Exim nie używa powłoki do uruchamiania polecenia. Zamiast tego dzieli ciąg na nazwę polecenia i argumenty i uruchamia je bezpośrednio. Wiadomość jest przekazywana jako standardowe wejście takiego polecenia.

Na przykład, aby przekierować listę pocztową do lokalnej grupy dyskusyjnej, mógłbyś użyć skryptu powłoki *gateit* i skonfigurować lokalny alias tak, by dostarczał wszystkie wiadomości z tej listy do skryptu za pomocą `|gateit`. Jeśli wiersz poleceń zawiera przecinek, należy go ująć w cudzysłów wraz z symbolem potoku.

* Opis ten został uproszczony. Można sprawić, by moduły zarządzające przekazały adresy do modułów transportowych, które dostarczą wiadomości do hostów zdalnych. I podobnie, rutery mogą przekazać adresy do lokalnego modułu transportowego, który zapisze wiadomość do pliku lub potoku. Możliwe jest także, by rutery w pewnych warunkach przekazały adresy do programów zarządzających.

Użytkownicy lokalni

Adres lokalny zwykle jest jednoznaczny ze skrzynką pocztową. Znajduje się ona przeważnie w katalogu */var/spool/mail* i nosi nazwę użytkownika, który jest również właścicielem pliku. Jeżeli plik nie istnieje, Exim go tworzy.

W pewnych konfiguracjach grupa jest ustawiana na taką, do której należy użytkownik, a tryb praw dostępu na 0600. W tych przypadkach procesy dostarczania działają z prawami użytkownika i użytkownik może usunąć całą skrzynkę. W innych konfiguracjach skrzynka pocztowa należy do grupy **mail** i ma prawo dostępu 0660. Procesy dostarczające działają z uid systemu i grupą **mail**, a użytkownicy nie mogą usuwać plików swoich skrzynek, choć mogą je opróżniać.

Zauważ, że choć katalog */var/spool/mail* jest obecnie standardowym miejscem umieszczania plików skrzynek pocztowych, niektóre programy są skompilowane do używania innych ścieżek, na przykład */usr/spool/mail*. Jeżeli dostarczenie poczty do użytkowników na twoim komputerze regularnie się nie udaje, powinieneś zobaczyć, czy pomoże stworzenie dowiązania symbolicznego do */var/spool/mail*.

Adresy **MAILER-DAEMON** i **postmaster** normalnie powinny być umieszczone w pliku aliasów i powinny się rozwijać do adresów e-mail administratora systemu. **MAILER-DAEMON** jest używany przez Exima jako adres nadawcy w wiadomościach odbitych. Jest również zalecane, by **root** był skonfigurowany jako alias dla administratora, szczególnie gdy dostarczanie odbywa się z prawami odbiorców, aby zapobiec dostarczaniu jako **root**.

Przekierowywanie poczty

Użytkownicy mogą przekierowywać swoją pocztę na inne adresy, tworząc plik *.forward* w swoich katalogach macierzystych. Zawiera on listę odbiorców, w której znakiem separatora jest przecinek i/lub znak nowego wiersza. Wszystkie wiersze zawarte w pliku są odczytywane i interpretowane. Można w nim użyć adresu dowolnego typu. Praktycznym przykładem pliku *.forward* przygotowanego na czas urlopu może być:

```
janet, "|vacation"
```

W innych opisach plików *.forward* możesz znaleźć nazwę użytkownika poprzedzoną znakiem odwrotnego ukośnika. W starszych MTA taki zapis zapobiegał szukaniu nowej nazwy w pliku *.forward*, co mogło prowadzić do zapętlenia. W Eximie odwrotny ukośnik nie jest potrzebny, gdyż program ten automatycznie rozwiązuje problem zapętlenia*. Jednak znak odwrotnego ukośnika jest dopuszczalny i nie jest on bez znaczenia w konfiguracji, obsługującej kilka domen naraz. Sama nazwa użytkownika, bez znaku odwrotnego ukośnika, jest uznawana za nazwę z domeny domyślnej. W przypadku zastosowania odwrotnego ukośnika zachowywana jest podana domena.

* Program zarządzający jest pomijany, jeżeli adres, który ma zostać przetworzony, jest taki sam jak adres użyty do jego wygenerowania.

Pierwszy adres w pliku przekierowania odpowiada za dostarczenie przychodzącej wiadomości do skrzynki pocztowej **janet**, natomiast polecenie *vacation* zwraca do nadawcy krótką informację*.

Poza obsługą „tradycyjnych” plików przekierowania, Exima można skonfigurować do pracy z bardziej skomplikowanymi plikami, zwanymi *filtrami*. Zamiast listy adresów, na które należy przekierować wiadomość, plik filtru może zawierać testy zawartości przychodzącej wiadomości, tak by na przykład wiadomość mogła być przekazana tylko wtedy, gdy temat zawiera hasło „pilne”. Administrator systemu musi zdecydować, czy wolno pozwolić użytkownikom na taką elastyczność.

Pliki aliasów

Exim może obsługiwać pliki aliasów kompatybilne z plikami *sendmaila*. Wpisy w pliku aliasów mogą mieć następującą postać:

```
alias: odbiorcy
```

`odbiorcy` to lista oddzielonych przecinkami adresów, którymi zostanie zastąpiony alias. Lista odbiorców może ciągnąć się przez kilka wierszy, jeżeli następny wiersz rozpoczyna się od białego znaku.

Specjalna funkcja pozwala Eximowi obsługiwać listy pocztowe, które są umieszczone niezależnie od pliku aliasów: jeżeli podasz jako odbiorcę `:include:`*`nazwa-pliku`*, Exim odczytuje zadany plik i zastępuje jego zawartość listą odbiorców. Alternatywa dla takiej obsługi list pocztowych jest opisana w następnym podrozdziale, *Listy pocztowe*.

Główny plik aliasów to */etc/aliases*. Jeżeli przyznałeś prawa zapisu do tego pliku grupie lub wszystkim, Exim odmówi jego użycia i wstrzyma przyjmowanie poczty lokalnej. Możesz jednak kontrolować test związany ze sprawdzaniem praw dostępu, ustawiając *modemask* w programie zarządzającym *system_aliases*.

Oto przykładowy plik *aliases*:

```
# plik /etc/aliases dla vbrew.com
hostmaster: janet
postmaster: janet
usenet: phil
# Lista pocztowa development.
development: joe, sue, mark, biff,
   /var/mail/log/development
owner-development: joe
# Og□oszenia ogólne s□ wysy□ane do ca□ego personelu.
announce: :include: /etc/Exim/staff,
   /var/mail/log/announce
owner-announce: root
# przej□cie z listy pocztowej ppp na lokaln□ grup□ dyskusyjn□
ppp-list: "|/usr/local/bin/gateit local.lists.ppp"
```

* Jeżeli zdecydujesz się na użycie programu *vacation*, upewnij się, że nie będzie on odpowiadał na wiadomości pochodzące z list pocztowych! Naprawdę można się zdenerwować, jeśli z każdą wiadomością z listy pocztowej dostaje się informację o czyimś urlopie. Administratorzy list pocztowych: jest to dobry przykład, że nie należy ustawiać pola `Reply-To:` w wiadomościach wysyłanych z grupy, na adres odbiorców listy.

Gdy w plikach aliasów znajdują się nazwy plików i polecenia w potoku, tak jak w powyższym przykładzie, Exim musi wiedzieć, pod jakim użytkownikiem mają działać programy dostarczające. Opcja *user* w pliku konfiguracyjnym Exima (a także *group*) musi być ustawiona dla programu zarządzającego, który obsługuje aliasy, albo dla modułów transportowych, na które są przekierowywane wiadomości.

Jeżeli w czasie dostarczania wiadomości na adres wygenerowany z pliku *aliases* wystąpi błąd, Exim jak zwykle wyśle do nadawcy wiadomość odbitą, o ile za pomocą opcji *errors_to* nie określisz, że odbite wiadomości mają być wysyłane do kogo innego, na przykład do postmastera.

Listy pocztowe

Zamiast pliku *aliases*, program zarządzający *forwardfile* może obsługiwać także listy pocztowe. Są one zwykle przechowywane w jednym katalogu, jak */etc/exim/lists/*, a lista o nazwie `nag-bugs` jest opisana na plikiem *lists/nag-bugs*. Plik ten powinien zawierać adresy członków listy oddzielone przecinkami lub znakami nowego wiersza. Wiersze rozpoczynające się od znaku # są traktowane jako komentarze. Prosty program zarządzający wykorzystujący te dane może wyglądać następująco:

```
lists:
   driver = forwardfile
   file = /etc/exim/lists/${local_part}
   no_check_local_user
   errors_to = ${local_part}-request
```

Gdy działa program zarządzający, wartości opcji *`file`* i *`errors_to`* są *rozwijane*. Rozwinięcie powoduje, że te fragmenty ciągu znaków, które rozpoczynają się od znaku dolara, zostaną za każdym razem zastąpione użytym ciągiem. Najprostszym rodzajem rozwinięcia jest wstawienie wartości jednej ze zmiennych Exima i tak właśnie się tutaj dzieje. Ciąg `${local_part}` jest zastępowany wartością `$local_part`, która jest lokalną częścią przetwarzanego adresu.

W każdej liście pocztowej powinien znajdować się użytkownik (lub alias) o nazwie *`listname-request`*. Wszelkie błędy występujące przy rozwiązywaniu adresu lub dostarczaniu poczty do członka listy są zgłaszane na ten adres.

Ochrona przed spamem

Spam, lub inaczej niechciana poczta reklamowa, jest problemem denerwującym wielu użytkowników. Do prac nad rozwiązaniem tego problemu powołano projekt MAPS (*Mail Abuse Protection System*). Stworzono też mechanizm zmniejszający skalę problemu, tak zwaną czarną listę (*Real Time Blackhole List* – RBL). Informacje o tym, jak działa RBL projektu MAPS, możesz znaleźć w dokumentacji elektronicznej pod adresem *http://maps.vix.com/rbl/*. Pomysł jest prosty. Ośrodki, które zostaną złapane na generowaniu spamu, są dodawane do bazy danych, a agenty przesyłające pocztę, takie jak Exim, są w stanie zadawać do tej bazy zapytania i sprawdzać przed przyjęciem poczty, czy host źródłowy nie jest spammerem.

Oprócz RBL, powstało już kilka innych podobnych list. Jedna z najbardziej użytecznych to DUL (*Dial-Up List*), zawierająca adresy IP hostów podłączonych przez linie komutowane. Normalnie powinny one wysyłać pocztę wychodzącą tylko przez serwery pocztowe swoich dostawców. Wiele ośrodków blokuje przyjmowanie poczty z zewnętrznych hostów komutowanych, ponieważ, gdy taki host nie używa serwera własnego dostawcy Internetu, zwykle nie wróży to nic dobrego.

Exim obsługuje różne czarne listy. Bardzo łatwo jest je w nim skonfigurować. Aby włączyć sprawdzanie takich list, dodaj poniższy wiersz do pliku */etc/exim.conf*:

```
# Vixie / MAPS RBL (http://maps.vix.com/rbl)
rbl_domains = rbl.maps.vix.com : dul.maps.vix.com
```

Ten przykład sprawdza zarówno RBL, jak i DUL, i odrzuca wszelkie wiadomości pochodzące z hostów, które znajdują się na którejkolwiek z list. Opcja *rbl_hosts* pozwala na podanie grupy hostów, których dotyczy (lub nie dotyczy) sprawdzanie RBL. Domyślne ustawienie jest następujące:

```
rbl_hosts = *
```

co oznacza, że wszystkie hosty są sprawdzane przez RBL. Gdybyś chciał wyłączyć sprawdzanie czarnej listy i przyjmować pocztę z danego hosta bez kontroli, mógłbyś na przykład zrobić następujący wpis:

```
rbl_hosts = ! nocheck.example.com : *
```

Wykrzyknik przed pierwszym elementem listy powoduje jej zanegowanie. Gdyby host łączący się z nami był *nocheck.example.com*, pasowałby do tego wyrażenia. Ale ze względu na negację, nie jest wykonywane sprawdzanie RBL. Wszelkie inne hosty pasują do drugiego elementu listy.

Konfigurowanie UUCP

Exim nie zawiera żadnego szczególnego kodu do wysyłania poczty przez UUCP ani nie obsługuje adresów w postaci wykazu trasowania UUCP. Jednak, jeżeli zostanie użyte adresowanie domenowe, Exim może bardzo łatwo stać się interfejsem dla UUCP. Oto, wzięty z rzeczywistej instalacji, fragment konfiguracji pozwalającej na wysyłanie pewnych domen do UUCP:

```
# Transport
uucp:
   driver = pipe
   user = nobody
   command = "/usr/local/bin/uux -r - \
     ${substr_-5:$host}!rmail ${local_part}"
   return_fail_output = true

# Router
uucphost:
   transport = uucp
   driver = domainlist
   route_file = /usr/exim/uucphosts
   search_type = lsearch
```

W kompletnym pliku konfiguracyjnym konfiguracja transportu zostałaby umieszczona wśród innych konfiguracji transportu, a ruter zostałby prawdopodobnie zdefiniowany jako pierwszy ruter. Plik */usr/exim/uucphosts* zawiera następujące wpisy:

```
darksite.example.com:    darksite.UUCP
```

które są interpretowane następująco: „Wyślij pocztę adresowaną do domeny **darksite.example.com** do hosta UUCP **darksite**". Ta konfiguracja mogłaby być zrealizowana prościej bez rutera dostawiającego przyrostek. UUCP do **darksite**, ale ten sposób jest przydatny, ponieważ pozwala odróżnić domenę **darksite.example.com** od nazwy hosta UUCP **darksite**.

Kiedy tylko ruter dotrze do domeny, która jest wpisana w pliku, przekazuje adres do transportu UUCP, który z kolei przekazuje go przez potok do polecenia *uux* (opisanego w rozdziale 16, *Zarządzanie UUCP Taylora*). Jeżeli napotka problem, *uux* wygeneruje jakiś wynik i zakończy działanie z niezerowym kodem błędu. Ustawienie zmiennej `return_fail_output` powoduje, że komunikat błędu zostaje zwrócony do nadawcy.

Jeżeli przychodzące wiadomości UUCP są grupowane w plikach we wsadowym formacie SMTP, mogą być przekazane bezpośrednio do Exima za pomocą poniższego polecenia:

```
exim -bS </var/uucp/incoming/001
```

Jednak jest tu jedna pułapka. Gdy Exim odbierze wiadomość lokalnie, nadawca musi być zalogowanym użytkownikiem, który go wywołał. W przypadku UUCP chcemy jednak, by nadawcy byli brani z przychodzących wiadomości. Exim to zrobi, jeżeli proces go wywołujący działa jako *użytkownik zaufany*. Jeżeli przychodząca poczta UUCP będzie obsługiwana na przykład przez użytkownika **uucp**, musisz w pliku konfiguracyjnym Exima wpisać:

```
trusted_users = uucp
```

Taki wpis zapewni poprawne obsłużenie adresów nadawców.



20

Grupy dyskusyjne

Grupy dyskusyjne Usenetu są jedną z najważniejszych i wysoko cenionych usług w dzisiejszych sieciach komputerowych. Choć niektórzy uważają Usenet za grzęzawisko agresywnej poczty komercyjnej i pornografii, są tam i doskonałe grupy dyskusyjne, które stanowiły niezastąpione źródło informacji, zanim pojawiło się WWW. Nawet w czasach bilionа stron WWW, grupy dyskusyjne pozostają miejscem, gdzie możesz znaleźć pomoc i podyskutować na wiele tematów.

Historia Usenetu

Pomysł grup dyskusyjnych zrodził się w 1979 roku, kiedy dwóch absolwentów, Tom Truscott i Jim Ellis, pomyślało o użyciu UUCP do połączenia komputerów w celu wymiany informacji pomiędzy użytkownikami Uniksa. Zbudowali oni w Karolinie Północnej małą sieć, składającą się z trzech komputerów.

Na początku ruch był obsługiwany przez kilka skryptów (później przepisanych w języku C), ale nigdy nie zostały one rozpowszechnione publicznie. Szybko zastąpiono je przez „A News” – pierwsze publicznie dostępne oprogramowanie dla grup dyskusyjnych.

A News mogło obsłużyć najwyżej kilka artykułów dziennie. Gdy liczba nadsyłanych do grupy artykułów zaczęła rosnąć, Mark Horton i Matt Glickman przepisali oprogramowanie i nazwali je wydaniem „B” (lub B News). Pierwsze publicznie dostępne wydanie B News miało numer wersji 2.1 i ujrzało światło dzienne w 1982 roku. Nieustannie było udoskonalane i wzbogacane o nowe funkcje. Aktualna wersja B News ma numer 2.11. Oprogramowanie to powoli przechodzi do historii, a ostatnia osoba, która je utrzymywała, zajęła się rozwojem programu INN.

Geoff Collyer i Henry Spencer przepisali B News i wydali je w 1987 roku jako wersję C (C News). Od czasu tego wydania pojawiło się szereg łat do C News, z których najbardziej wartościowe było wydanie C News Performance Release. W ośrodkach obsługujących wiele grup są dosyć duże obciążenia związane z częstym wywoływa-

niem *relaynews*, które odpowiada za rozdzielanie przychodzących artykułów do innych hostów. Wersja Performance dodaje do *relaynews* kilka opcji, które pozwalają działać programowi w *trybie demona* w tle. Wersja Performance C News jest aktualnie załączana do większości wydań Linuksa. C News szczegółowo opisujemy w rozdziale 21, *C News*.

Wszystkie wersje, aż do C, były pisane z myślą o sieci UUCP, choć mogły być również stosowane w innych środowiskach. Efektywne przesyłanie wiadomości przez sieci, takie jak TCP/IP czy DECNet, wymagało nowej architektury. Dlatego w 1986 roku został wymyślony *protokół przesyłania wiadomości w sieci komputerowej Usenet* (*Network News Transfer Protocol* – NNTP). Jest on oparty na połączeniach sieciowych i zawiera szereg poleceń do interaktywnego przesyłania i odbierania artykułów.

W sieci można znaleźć wiele aplikacji opartych na NNTP. Jedną z nich jest pakiet *nntpd*, stworzony przez Briana Barbera i Phila Lapsleya. Służy on do udostępniania grup dyskusyjnych hostom w sieci lokalnej. W założeniu *nntpd* miał uzupełniać pakiety oprogramowania obsługującego grupy dyskusyjne, czyli B News lub C News. Dodaje do nich funkcje NNTP. Jeśli chcesz używać NNTP z serwerem C News, powinieneś przeczytać rozdział 22, *NNTP i demon nntpd*, wyjaśniający, jak skonfigurować demona *nntpd* i uruchomić go z C News.

Alternatywnym pakietem obsługującym NNTP jest INN lub *InternetNews*. Nie jest to jedynie interfejs, ale system grup działający na własnych prawach. Składa się z wyrafinowanego demona przekazującego dane, który efektywnie obsługuje kilka jednoczesnych połączeń NNTP, oraz z serwera grup używanego w wielu ośrodkach w Internecie. Szczegółowo omawiamy go w rozdziale 23, *Internet News*.

Czym jest Usenet

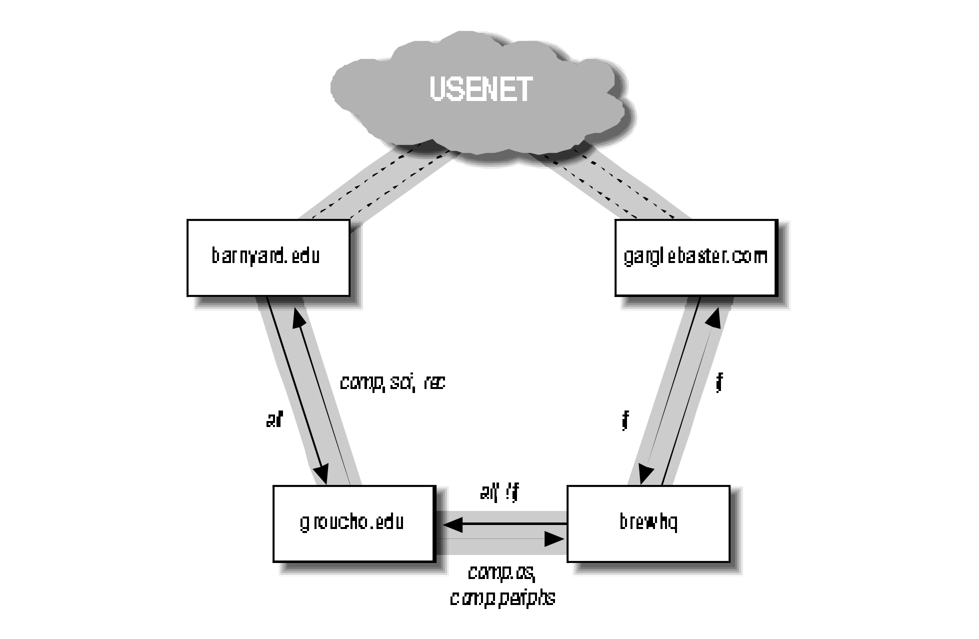
Jednym z bardziej zdumiewających faktów związanych z Usenetem jest to, że nie jest on częścią organizacji, ani nie ma żadnej centralnej władzy zarządzającej. W zasadzie taki już jest Usenet, że poza opisem technicznym nie da się go zdefiniować. Ryzykując, że będzie to brzmiało śmiesznie, można zdefiniować Usenet jako współpracę ośrodków wymieniających wiadomości grup dyskusyjnych Usenetu. Aby stać się ośrodkiem Usenetu, wystarczy znaleźć inny ośrodek Usenetu i uzgodnić z jego właścicielami sposób i prawa wymiany wiadomości grup dyskusyjnych między wami. Udostępnianie wiadomości grup dyskusyjnych innym ośrodkom nosi w języku angielskim nazwę *feeding* (dosłownie: *karmienie*). W dalszej części tej książki będziemy używali terminu *dostarczanie*.

Podstawową jednostką grup dyskusyjnych Usenetu jest *artykuł*. Jest to wiadomość, napisana przez użytkownika i „wysłana" do sieci. Aby system grup dyskusyjnych mógł ją obsłużyć, dodawane są do niej informacje administracyjne (tak zwany nagłówek artykułu). Jest on bardzo podobny do nagłówka poczty zgodnego ze standardem RFC-822. Również składa się z kilku wierszy tekstu, z których każdy rozpoczyna się od nazwy pola zakończonej dwukropkiem; po nim występuje wartość pola.*

* Format wiadomości Usenet jest określony przez RFC-1036 *Standard for interchange of USENET messages*

Artykuły są wysyłane do jednej lub kilku *grup dyskusyjnych*. Można powiedzieć, że grupa dyskusyjna to forum artykułów związanych z jakimś tematem. Wszystkie grupy dyskusyjne są uporządkowane w pewnej hierarchii, a nazwa grupy wskazuje jej miejsce w tej hierarchii. Często na tej podstawie łatwo jest stwierdzić, czego dotyczy dana grupa. Na przykład każdy może na podstawie nazwy grupy *comp.os.linux.announce* stwierdzić, że dotyczy ona ogłoszeń związanych z komputerowym systemem operacyjnym o nazwie Linux.

Artykuły te są następnie wymieniane pomiędzy wszystkimi ośrodkami Usenetu, które chcą udostępniać daną grupę. Gdy dwa ośrodki ustalą, że będą wymieniać wiadomości, mogą przesyłać sobie dowolne grupy, a nawet dodawać swoje lokalne hierarchie grupy. Na przykład **groucho.edu** mógłby mieć połączenie z **barnyard.edu**, czyli głównym dostawcą grup, i kilka połączeń z mniejszymi ośrodkami, do których by te grupy przekazywał. Barnyard College może odbierać wszystkie grupy Usenetu, natomiast GMU tylko kilka głównych hierarchii, jak *sci*, *comp* czy *rec*. Jakiś inny ośrodek, powiedzmy ośrodek UUCP **brewhq**, będzie udostępniać jeszcze mniej grup, ponieważ nie ma wystarczających zasobów sieciowych i sprzętowych na obsługę wszystkich. Z drugiej strony **brewhq** może jednak obsługiwać grupy z hierarchii *fj*, których nie ma GMU. Musi więc mieć dodatkowe połączenie z **garglebaster.com**, który posiada wszystkie grupy *fj* i dostarcza je do **brewhq**. Przepływ grup pokazuje rysunek 20-1.



Rysunek 20-1. Przepływ grup Usenet na uniwersytecie Groucho Marx

Podpisy przy strzałkach pochodzących od **brewhq** mogą jednak wymagać pewnego wyjaśnienia. Domyślnie, **brewhq** chce, aby wszystkie grupy generowane lokalnie były wysyłane do **groucho.edu**. Jednak ponieważ **groucho.edu** nie obsługuje grup *fj*, nie ma sensu wysyłać żadnych wiadomości z tych grup. Dlatego dane przesyłane z **brewhq** do GMU są opisane jako: `all, !fj`, co oznacza, że są wysyłane wszystkie grupy poza *fj*.

Jak Usenet obsługuje grupy dyskusyjne

W dzisiejszych czasach Usenet rozrósł się do niesłychanych rozmiarów. Ośrodki posiadające wszystkie grupy zwykle przesyłają marne 60 MB dziennie*. Oczywiście nie da się tego zrobić zwykłym rozdaniem plików do okoła. Przyjrzyjmy się więc, w jaki sposób większość systemów Unix obsługuje grupy Usenet.

Wszystko zaczyna się, gdy użytkownicy piszą i wysyłają artykuły. Każdy użytkownik pisze artykuły w specjalnej aplikacji, tak zwanej przeglądarce grup dyskusyjnych (ang. *newsreader*), która odpowiednio je formatuje w celu przesłania do lokalnego serwera grup dyskusyjnych. W środowiskach uniksowych przeglądarka grup zwykle używa polecenia *inews* do przesłania artykułów do serwera za pomocą protokołu TCP/IP. Możliwe jest jednak również zapisanie artykułu bezpośrednio do pliku w specjalnym katalogu nazywanym buforem grup. Gdy tak przygotowana wiadomość zostanie dostarczona do lokalnego serwera grup dyskusyjnych, bierze on odpowiedzialność za dostarczenie artykułu do innych użytkowników grupy.

Grupy są rozpowszechniane w sieci za pomocą różnych protokołów transportowych. Kiedyś najczęściej korzystano z UUCP, ale obecnie główny ruch jest generowany przez ośrodki internetowe. Używany algorytm rutingu jest nazywany *trasowaniem rozpływowym* (ang. *flooding*). Każdy ośrodek utrzymuje kilka połączeń (*dostawców grup* – ang. *news feeds*) z innymi ośrodkami. Każdy artykuł wygenerowany lub odebrany przez lokalny system grup jest przekazywany ośrodkom, o ile jeszcze w nich nie był. Ośrodek może dowiedzieć się, w jakich ośrodkach artykuł już był, odczytując pole nagłówka `Path:`. Nagłówek ten zawiera listę wszystkich systemów, przez które artykuł przechodził, zapisaną w notacji wykazu trasowania.

Aby rozróżnić artykuły i wykryć duplikaty, artykuły Usenet mają identyfikatory (ID) wiadomości (określone w polu nagłówka `Message-ID:`), które składają się z nazwy ośrodka wysyłającego go i numeru seryjnego go: *<numer@o□rodek>*. ID każdego przetworzonego artykułu jest zapisywane w pliku *history*, z którym są porównywane wszystkie nowo przychodzące artykuły.

Przepływ pomiędzy dwoma dowolnymi ośrodkami może być ograniczony przez dwa kryteria. Z jednej strony nadawca przypisuje artykułowi dystrybucję (w polu nagłówka `Distribution:`). W ten sposób można zawsze rozpowszechnić artykuł do określonej grupy ośrodków. Z drugiej strony również system odbiorczy może nałożyć swoje ograniczenia. Zestaw grup dyskusyjnych i dystrybucji, które mogą być przesyłane przez ośrodki, najczęściej jest opisany w pliku *sys*.

---

* Zaraz, zaraz... 60 MB z prędkością 9600 bps, to daje 60 milionów razy 1024, czyli... jakieś 34 godziny!

Zwykle potrzebne są jakieś poprawki w tym schemacie. W sieciach UUCP systemy zbierają artykuły przez jakiś czas, łączą je w jeden plik, który jest kompresowany i wysyłany do ośrodka zdalnego. Procedura ta nosi nazwę *przetwarzania wsadowego* (ang. *batching*).

Alternatywną techniką jest protokół *ihave/sendme*, który zapobiega przesyłaniu zduplikowanych artykułów, dzięki czemu oszczędza przepustowość sieci. Zamiast umieszczać wszystkie artykuły w plikach wsadowych i wysyłać je w całości, w gigantycznym pliku „ihave” łączone są tylko ID wiadomości i wysyłane do ośrodka zdalnego. Ośrodek zdalny odczytuje ten plik, porównuje z plikiem historii, po czym w wiadomości „send me” zwraca listę artykułów, których potrzebuje. Wysyłane są tylko żądane artykuły.

Oczywiście protokół ihave/send me ma sens tylko wtedy, gdy dotyczy dwóch dużych ośrodków, które pobierają grupy z niezależnych źródeł i sprawdzają się nawzajem na tyle często, żeby przepływ wiadomości był efektywny.

Ośrodki w Internecie zwykle opierają się na oprogramowaniu TCP/IP, które wykorzystuje protokół NNTP (*Network News Transfer Protocol*). NNTP jest opisany w RFC-977. Jest on odpowiedzialny za przesyłanie grup między serwerami i zapewnia pojedynczym użytkownikom dostęp do zdalnych hostów.

NNTP oferuje trzy różne sposoby przesyłania grup. Jeden to wersja ihave/send me działająca w czasie rzeczywistym, nazywana także *wciskaniem* (ang. *pushing*) grup. Druga technika nosi nazwę *ściągania* (ang. *pulling*) grup i w niej klient prosi o listę artykułów w danej grupie lub hierarchii, które dotarły do serwera po określonym czasie, i wybiera te, których nie ma w pliku historii. Trzecia technika służy do interaktywnego czytania grup i pozwala tobie lub twojej przeglądarce grup na pobieranie artykułów z zadanych grup oraz wysyłanie artykułów z niepełną informacją w nagłówku.

W każdym ośrodku grupy dyskusyjne są przechowywane w hierarchii katalogów poniżej */var/spool/news*, gdzie każdy artykuł znajduje się w oddzielnym pliku, a każda grupa w oddzielnym katalogu. Nazwa katalogu jest budowana na podstawie nazwy grupy, z tym że poszczególne człony są kolejnymi podkatalogami. I tak, artykuły z *comp.os.linux.misc* są przechowywane w */var/spool/news/comp/os/linux/misc*. Artykułom w grupie są przypisywane numery w kolejności, w jakiej artykuły nadchodzą. Tymi numerami nazywane są kolejne pliki. Zakres numerów aktualnie dostępnych artykułów jest przechowywany w pliku *active*, który służy jednocześnie jako lista artykułów znanych danemu ośrodkowi.

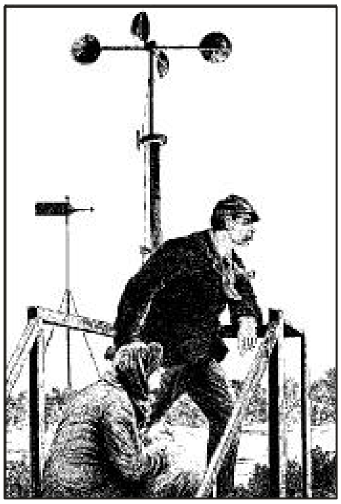
Ponieważ miejsca na dysku stopniowo ubywa, musisz po jakimś czasie wyrzucać artykuły*. Zwykle artykuły z pewnych grup i hierarchii wygasają po określonej liczbie dni od ich przybycia. Może to zmienić autor, określając datę wygaśnięcia w polu `Expires:` nagłówka artykułu.

* Niektórzy uważają, że Usenet jest spiskiem producentów modemów i dysków twardych. Nazywa się to *wygasaniem* (ang. *expiring*)

Wiesz już wystarczająco dużo, by samemu wybrać sobie dalsze lektury. Użytkownicy UUCP powinni przeczytać rozdział 21 dotyczący C News. Jeżeli korzystasz z sieci TCP/IP, przeczytaj rozdział 22 omawiający NNTP. Jeżeli chcesz przesyłać umiarkowaną liczbę grup przez TCP/IP, serwer tam opisany może ci wystarczyć. Aby zainstalować wydajny serwer grup dyskusyjnych, który jest w stanie obsługiwać ogromną liczbę materiału, przeczytaj rozdział 23, *Internet News*.

21

C News



Jednym z najpopularniejszych pakietów oprogramowania grup dyskusyjnych jest C News. Został zaprojektowany dla ośrodków obsługujących grupy dyskusyjne przez łącza UUCP. Ten rozdział omawia ogólne pojęcia C News, podstawową instalację i zadania administracyjne.

C News przechowuje swoją konfigurację w plikach w katalogu */etc/news*, a większość jego plików binarnych znajduje się w katalogu */usr/lib/news*. Artykuły są przechowywane w katalogu */var/spool/news*. Powinieneś zadbać o to, aby praktycznie wszystkie pliki w tych katalogach były własnością użytkownika **news** lub grupy **news**. Problemy powstają głównie wtedy, gdy pliki są niedostępne dla C News. Użyj polecenia *su*, by stać się użytkownikiem **news**, zanim zaczniesz cokolwiek robić z tymi katalogami. Jedynym wyjątkiem jest polecenie *setnewsids*, używane do ustawienia rzeczywistego ID użytkownika niektórych programów do obsługi grup dyskusyjnych. Musi być ono własnością użytkownika **root** i mieć ustawiony bit setuid.

W tym rozdziale opiszemy szczegółowo wszystkie pliki konfiguracyjne C News i pokażemy, co musisz zrobić, by twój ośrodek działał.

Dostarczanie grup dyskusyjnych

Artykuły mogą być dostarczane do C News na kilka sposobów. Gdy lokalny użytkownik wysyła artykuł, przeglądarka grup dyskusyjnych zwykle przekazuje go poleceniem *inews*, które uzupełnia informacje w nagłówku. Grupy z ośrodków zdalnych, czy to pojedyncze artykuły, czy całe pliki wsadowe, są przekazywane poleceniem *rnews*, które zapisuje je w katalogu */var/spool/news/in.coming*, z którego z kolei

są później pobierane przez *newsrun*. W każdej z obu tych technik artykuł ostatecznie został nie przekazany do polecenia *relaynews*.

Polecenie *relaynews* najpierw sprawdza, czy artykuł był już w ośrodku lokalnym. W tym celu przegląda ID wiadomości w pliku *history*. Zduplikowane artykuły są odrzucane. Następnie *relaynews* zagląda do pola `Newsgroups:` nagłówka, aby dowiedzieć się, czy lokalny ośrodek przyjmuje artykuły z tych grup. Jeżeli tak, a grupa jest wpisana w pliku *active*, *relaynews* próbuje zachować artykuł w odpowiednim katalogu w obszarze bufora grup. Jeżeli katalog nie istnieje, jest tworzony. ID artykułu jest następnie zapisywane do pliku *history*. W przeciwnym razie *relaynews* odrzuca artykuł.

Czasem poleceniu *relaynews* nie uda się zachować przychodzącego artykułu, ponieważ grupa, do której został wysłany, nie istnieje w twoim pliku *active*. W takiej sytuacji, artykuł jest przenoszony do grupy *junk**; *relaynews* sprawdza także, czy artykuł nie jest stary lub źle datowany. Jeśli jest – odrzuca go. Przychodzące wsady, które mają jakiekolwiek błędy, są przenoszone do katalogu */var/spool/news/in.coming/bad* i zapisywany jest komunikat błędu.

Następnie artykuł jest przekazywany do wszystkich pozostałych ośrodków, które żądały wiadomości z tych grup. W tym celu korzysta się ze środka transportu ośrodka zdalnego. Aby artykuł nie został wysłany do ośrodka, w którym już był, każdy docelowy ośrodek jest sprawdzany w polu nagłówka `Path:`, które zawiera listę ośrodków, przez które artykuł do tej pory przeszedł, zapisanych w postaci wykazu trasowania UUCP (patrz rozdział 17, *Poczta elektroniczna*). Jeżeli nazwy ośrodka docelowego nie ma na tej liście, artykuł jest do niego wysyłany.

System C News jest powszechnie używany do przekazywania grup dyskusyjnych pomiędzy ośrodkami UUCP, choć możliwe jest także jego zastosowanie w środowisku NNTP. Do dostarczenia wiadomości do zdalnego ośrodka UUCP, czy to będą pojedyncze artykuły, czy całe wsady, służy polecenie *uux*. Polecenie to uruchamia *rnews* w zdalnym ośrodku i przekazuje artykuł lub wsad na jego standardowe wejście. Więcej informacji na temat UUCP znajdziesz w rozdziale 16, *Zarządzanie UUCP Taylora*.

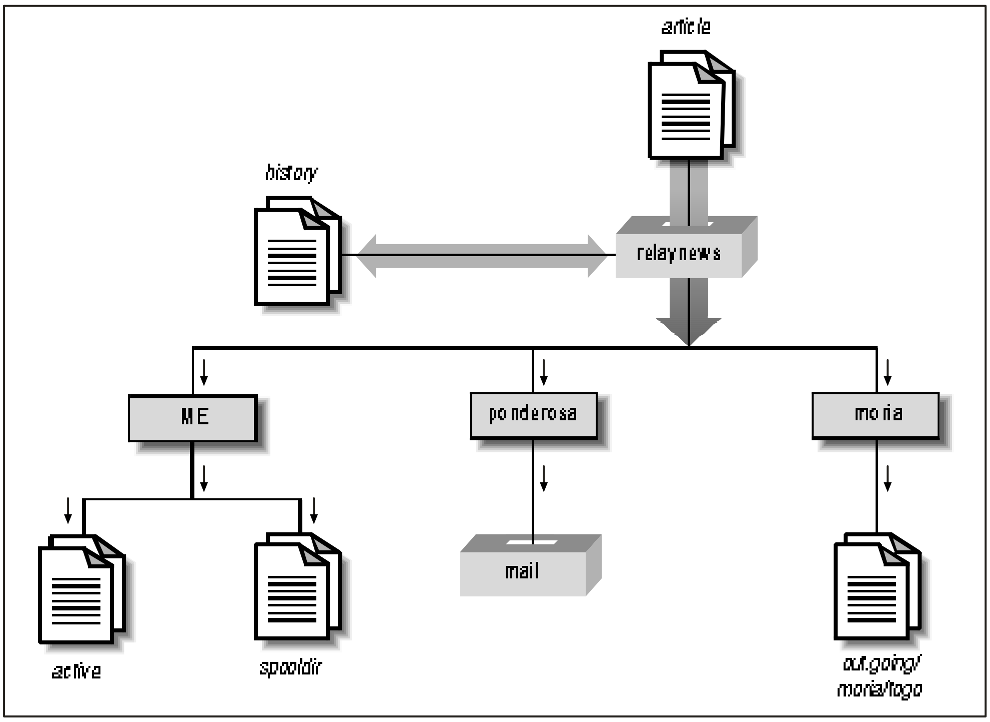
Przetwarzanie wsadowe (ang. *batching*) oznacza wysyłanie dużych porcji pojedynczych artykułów za jednym razem. Gdy przetwarzanie wsadowe jest włączone dla danego ośrodka, C News nie wysyła przychodzących artykułów natychmiast, ale dodaje ścieżkę do pliku, zwykle *out.going/site/togo*. Co jakiś czas z *crontaba* jest wykonywany program, który odczytuje pliki i pakuje wszystkie wskazane artykuły w jeden lub kilka plików, opcjonalnie je kompresuje i wysyła do *rnews* w ośrodku zdalnym**.

---

* Mogą istnieć różnice pomiędzy grupami obecnymi w twoim ośrodku, a tymi, które chce on otrzymywać. Na przykład na liście subskrypcyjnej może znajdować się *comp.all*, co oznacza przesyłanie wszystkich grup z hierarchii *comp*, ale twój ośrodek może nie zawierać w pliku *active* kilku z nich. Artykuły wysyłane do tych brakujących grup są przenoszone do *junk*.

** Zauważ, że powinien to być *crontab* użytkownika **news**. Wtedy nie zostaną pomieszane prawa dostępu do plików.

Rysunek 21-1 pokazuje przekierowywanie wiadomości przez *relaynews*. Artykuły mogą być przekazywane do ośrodka lokalnego (oznaczonego jako `ME`), do ośrodka o nazwie **ponderosa** przez e-mail i do ośrodka **moria**, dla którego włączone jest przetwarzanie wsadowe.



Rysunek 21-1. Przepływ wiadomości przez relaynews

Instalacja

C News powinien znajdować się w postaci pakietu we wszystkich obecnie dostępnych dystrybucjach Linuksa, tak więc instalacja jest dosyć prosta. Jeżeli go nie ma lub jeżeli chcesz instalować oryginalny kod źródłowy, możesz to oczywiście zrobić*. Bez względu na to, jak go zainstalujesz, będziesz musiał dokonać edycji plików konfiguracyjnych. Ich formaty są opisane poniżej:

sys
: Plik *sys* kontroluje, które grupy twój ośrodek otrzymuje i przekazuje dalej. Omawiamy go szczegółowo w następnym podrozdziale.

active
: Zwykle nie jest edytowany przez administratora. Zawiera wskazówki co do obsługi artykułów z grup dyskusyjnych obsługiwanych przez ośrodek.

* Pakiet z kodem źródłowym C News możesz uzyskać z ośrodka macierzystego **ftp.cs.toronto.edu** */pub/c-news/c-news.tar.Z*

organization
Ten plik po win ien za wier ać na zwę two jej or gan iza cji, na przykład „Bro war wir tua lny Inc.”. W swo im kom put erze wpisz „ośrod ek pry watny” lub co kolw iek innego. Wię k szość osób nie uzna two jego ośrodka za skon fig uro wany po prawn ie, je żeli nie bę dziesz miał tego pli ku.

newsgroups
Ten plik za wie ra li stę wszyst kich grup dys ku syj nych z jed no wier szo wym opi sem ka ż dej z nich. Te opi sy czę sto są uży wa ne przez przeglądar ki grup przy wy świetla niu li sty grup, do kt órych je steś za pi sa ny.

mailname
Na zwa pocz towa two jego ośrodka, na przykład **vbrew.com**.

whoami
Na zwa two jego ośrodka używ ana do celów związa nych z gru pami dys kus yjnymi. Czę sto używ ane są na zwy UUCP oś rodków, na przykład **vbrew**.

explist
Po wi nie neś ra czej wy edyt ować ten plik, umieszc zając w nim pre fer owa ne cza sy wy gaś nięcia dla da nych grup dys kus yjny ch. Miej sce na dys ku może od gryw ać istotną rolę w do kon ywa nych przez cie bie wy bor ach.

W celu utworzenia wstępnej hierarchii grup dyskusyjnych, zdobądź pli ki *active* i *newsgroups* z ośrod ka, z które go po bie rasz gru py. Za in sta luj je w ka ta lo gu */etc/news*, upew niając się, że są własno ścią użyt kow ni ka **news** i ustaw tryb 644 poleceniem *chmod*. Usuń z pli ku *active* wszyst kie gru py *to.** i do daj *to.moj-osrodek*, *to.osro dek-dostarczajacy*, *junk* i *control*. Gru py *to.** zwy kle są uży wa ne do wy mia ny wia do mo ści ihave/sendme. Powinieneś je mieć bez względu na to, czy planujesz używać ihave/send me, czy nie. Na stęp nie zmień wszyst kie nu me ry ar ty kułów w dru gim i trze cim po lu *active* za po mocą po niż szych po le ceń:

```
# cp active active.old
# sed 's/ [0-9]* [0-9]* / 0000000000 00001 /' active.old > active
# rm active.old
```

Dru gie po lec enie wy wołuje edyt or stru mien iowy *sed*. Wy wołanie to za stęp uje dwa ciągi znaków skład ające się z cyfr, od po wied nio, ciągiem zer i ciągiem `00001`.

Na ko niec st wórz ka tal og bu for owy grup dys kus yjny ch i pod kat alo gi używ ane dla przy chodzących i wy chodzących grup:

```
# cd /var/spool
# mkdir news news/in.coming news/out.going news/out.master
# chown -R news.news news
# chmod -R 755 news
```

Je żeli używ asz skom pil owa nej przeglądar ki grup po chodzącej z in nej dys tryb ucji C News niż ser wer, może się okaz ać, że oczek uje ona bu fora grup w ka tal ogu */usr/spool/news*, a nie w */var/spool/news*. Je żeli two ja przeglądar ka grup nie wi dzi żadn ych artykułów, st wórz do wiąza nie sym bol iczne od */usr/spool/news* do */var/spool/news* w następ ujący sposób:

```
# ln -sf /usr/spool/news /var/spool/news
```

Teraz jesteś gotów na przyjmowanie grup dyskusyjnych. Zauważ, że nie musisz two rzyć kat alo gów dla poszczególnych grup. C News au to ma tycz nie two rzy brakujące ka ta lo gi bu fo ro we dla wszyst kich grup, których ar ty kuły przyj mu je

W szcze gó lności są one two rzo ne dla *wszystkich* grup, do których ar ty kuł był wysyłany w sposób wie lo adre so wy (ang. *cross-posted*). Po chwi li stwier dzisz więc, że twój ka ta log bu fo ro wy wy pełnił się ka ta lo ga mi grup, do których nig dy się nie za pisywałeś, jak *alt.lang.teco*. Możesz temu zapobiec, usuwając wszystkie niechciane gru py z pli ku *active* lub re gu lar nie uru cha miając skrypt usu wający wszyst kie pu ste podkatalogi katalogu */var/spool/news* (oczy wiś cie za wyjątkiem *out.going* i *in.coming*).

C News potrzebuje użytkownika, do którego może wysyłać komunikaty błędów i ra por ty o sta nie. Do my śl nie jest nim **usenet**. Je żeli uży wasz usta wień do my śl nych, mu sisz stwo rzyć alias, dzię ki któr emu pocz ta bę dzie prze ka zy wa na do jed nej lub kilku odpowiedzialnych osób. Możesz także zmienić to zachowanie, ustawiając zmienną śro do wi skową `NEWSMASTER` na odpowiednią nazwę. Musisz to zrobić w pli ku *crontab* użyt kow ni ka **news**. To po stępo wa nie trze ba po wta rzać za ka żdym ra zem, gdy uru cha miasz ręcz nie na rzę dzia ad mi ni stra cyj ne, a więc za pew ne łatwiej będzie zainstalować alias. Aliasy pocztowe są opisane w rozdziale 18, *Sendmail*, i w roz dzia le 19, *Exim*.

Gdy edy tu jesz plik */etc/passwd*, za dbaj o to, by ka żdy użyt kow nik miał wpi sa ne swoje praw dzi we na zwi sko w po lu `pw_gecos` (czwar te po le). Zgod nie z ne ty kietą (etykietą działania w sieci), nazwisko rzeczywistego nadawcy powinno pojawić się w po lu `From:` ar ty kułu. Oczy wiś cie i tak bę dziesz chciał, że by to po le było po prawnie wy pełnio ne, je żeli uży wasz pocz ty.

Plik sys

Plik *sys* umieszczony w katalogu */etc/news* kontroluje, które hierarchie odbierasz i prze ka zu jesz da lej do in nych oś rodków. Choć ist nieją na rzę dzia zarządzające o nazwach *addfeed* i *delfeed*, wy da je nam się, że le piej jest utrzy my wać ten plik ręcz nie.

Plik *sys* za wie ra wpi sy dla ka ż de go ośrod ka, które mu prze ka zu jesz gru py, oraz opis grup, które przyj mu jesz. Pierw szy wiersz to wpis `ME` opi sujący twój sys tem. Bezpiecz nie jest za pi sać go tak:

```
ME:all/all::
```

Mu sisz ta kże do dać wiersz dla ka ż de go ośrod ka, które mu do star czasz gru py. Ka żdy wiersz wygląda tak:

```
o□rodek[/wykluczenia]:listagrup[/listadyst][:znaczniki[:polecenia]]
```

Wpi sy mogą ciągnąć się przez kil ka wier szy, je żeli użyj esz zna ku od wrotn ego ukośnika (\) na ko ńcu wier sza, który ma być kon tyn uow any. Znak hasha (#) wska zuje na komentarz.

`o□rodek`
: Jest to na zwa ośrodka, którego do tyc zy wpis. Zwy kle umieszc za się tu taj na zwę UUCP ośrodka. W pli ku *sys* musi znaj dow ać się ta kże wpis dla two jego ośrod ka; inaczej nie bę dziesz otrzym ywał ar tykułów.

Specjalna nazwa `ME` oznacza twój ośrodek. Wpis `ME` definiuje wszystkie grupy, które chcesz przechowywać lokalnie. Artykuły nie pasujące do wiersza `ME` będą przenoszone do grupy *junk*.

Aby nie dopuścić do powstania pętli, C News odmawia przyjęcia wszystkich artykułów, które przeszły już przez dany ośrodek. W tym celu sprawdza, czy lokalny ośrodek nie pojawił się w polu `Path:` artykułu. Niektóre ośrodki mogą być znane pod różnymi poprawnymi nazwami. Na przykład niektóre ośrodki używają w tym polu swoich pełnych nazw domenowych lub aliasu na przykład `news.`*`o□rodek.domena`*. Aby mieć pewność, że mechanizm zapobiegania powstawaniu pętli zadziała, ważne jest dodanie wszystkich aliasów do listy wykluczeń. Wpisuje się je, oddzielając przecinkami.

W przypadku wpisu dotyczącego ośrodka **moria**, pole *`o□rodek`* miałoby na przykład wartość `moria/moria.orcnet.org`. Jeżeli **moria** miałby również alias **news.orcnet.org**, to nasze pole *`o□rodek`* miałoby wartość `moria/moria.orcnet.org,news.orcnet.org`.

`listagrup`

Jest to, oddzielana przecinkami, lista grup, do których jesteśmy zapisani, i hierarchii dla danego ośrodka. Hierarchia może być podana przez określenie przedrostka (jak *comp.os* dla wszystkich grup, które się zaczynają od takiego ciągu słów), po którym opcjonalnie występuje słowo kluczowe *all* (czyli np. *comp.os.all*).

Możesz wykluczyć jakąś hierarchię lub grupę z przekazywania, poprzedzając ją wykrzyknikiem. Jeżeli grupa jest sprawdzana z listą, zawsze jest dopasowywana najdłuższym zgodnym ciągiem znaków. Na przykład gdyby *`listagrup`* zawierała taką listę:

```
!comp,comp.os.linux,comp.folklore.computers
```

to z hierarchii comp. zostałyby pobrane tylko *comp.folklore.computers* i grupy *comp.os.linux*.

Jeżeli ośrodek ma przekazywać wszystkie grupy, które sam pobiera, wprowadź jako *`listagrup`* słowo kluczowe `all`.

`listadyst`

Ta wartość jest oddzielona od *`listygrup`* ukośnikiem i zawiera listę dystrybucji do przekazywania. Znów możesz wykluczyć pewne dystrybucje, poprzedzając je wykrzyknikiem. Wszystkie dystrybucje są opisywane słowem `all`. Pominięcie *`listydyst`* powoduje przyjęcie wartości domyślnej `all`.

Na przykład możesz użyć listy dystrybucji: `all,!local` aby grupy przeznaczone tylko do użytku lokalnego nie były wysyłane do ośrodków zdalnych.

Zwykle istnieją co najmniej dwie dystrybucje: `world`, często stosowana domyślnie, gdy użytkownik nie wskaże innej, i `local`. Mogą istnieć inne dystrybucje dotyczące zadanego regionu, stanu, kraju itd. Poza tym istnieją dwie dystrybucje używane tylko przez C News. Są to: `sendme` i `ihave` wykorzystywane w protokole sendme/ihave.

Można się zastanawiać, czy warto używać dystrybucji. Pole dystrybucji w artykule może być tworzone losowo, ale aby dystrybucja działała, serwery grup w sieci muszą ją znać. Niektóre błędnie działające przeglądarki grup tworzą fałszywe dystrybucje, ponieważ zakładają, że sensowną dystrybucją jest główny poziom hierarchii artykułu, na przykład, że dla *comp.os.linux.networking* byłaby to *comp*. Dystrybucje dotyczące regionów także są często wątpliwe, ponieważ wiadomość może wyjść poza region, gdy jest wysyłana przez Internet*. Dystrybucje związane z firmą są jednak sensowne. Można je stosować, aby zapobiec wyciekowi tajnych informacji poza sieć firmową. Ten cel jednak lepiej jest osiągnąć, tworząc oddzielną grupę lub hierarchię.

`znaczniki`

Ta opcja opisuje pewne parametry wysyłanej porcji wiadomości. Może być pusta lub stanowić połączenie następujących znaczników:

- `F` Ten znacznik włącza przetwarzanie wsadowe.
- `f` Prawie identyczny z `F`, ale pozwala C News na dokładniejsze obliczenie rozmiaru wychodzących plików wsadowych i raczej ten znacznik powinien być używany zamiast `F`.
- `I` Ten znacznik powoduje, że C News generuje listę artykułów odpowiednią do użycia z ihave/sendme. Aby uruchomić protokół ihave/sendme, wymagane są dodatkowe modyfikacje w pliku *sys* i pliku *batchparms*.
- `n` Ten znacznik tworzy pliki wsadowe dla aktywnych klientów NNTP, jak *nntpxmit* (zobacz rozdział 22, *NNTP i demon nntpd*). Pliki wsadowe zawierają nazwę pliku z artykułem oraz jego ID.
- `L` Ten znacznik mówi C News, aby przesyłał tylko artykuły stworzone w twoim ośrodku. Po tym znaczniku można wpisać liczbę dziesiętną *n*, która powoduje, że C News wysyła artykuły tylko w obrębie *n* hopów od twojego ośrodka. C News określa liczbę hopów na podstawie pola `Path:`.
- `u` Ten znacznik mówi C News, aby przetwarzał wsadowo tylko artykuły z grup niemoderowanych.
- `m` Ten znacznik mówi C News, by przetwarzał wsadowo tylko artykuły z grup moderowanych.

Możesz użyć najwyżej jednego ze znaczników `F`, `f`, `I` lub `n`.

`polecenia`

To pole zawiera polecenie, które zostanie wykonane dla każdego artykułu, o ile nie włączysz przetwarzania wsadowego. Artykuł będzie przekazany na standardowe wejście polecenia. Ta opcja powinna być używana tylko przy małej liczbie artykułów. W przeciwnym razie obciążenie obu systemów będzie zbyt duże.

Domyślne polecenie to:

```
uux - -r -z system-zdalny!rnews
```

* Nie jest niczym dziwnym, że artykuł wysłany, powiedzmy, w Hamburgu, idzie do Frankfurtu przez **reston.asn.net** w Holandii lub nawet przez jakieś ośrodki w Stanach.

Wywołuje ono *rnews* w systemie zdalnym, przekazując artykuł na jego standardowe wejście.

Domyślna ścieżka poszukiwania zdefiniowana dla poleceń umieszczanych w tym polu to: */bin:/usr/bin:/usr/lib/news/batch*. Ten ostatni katalog zawiera skrypty powłoki, których nazwy zaczynają się od *via*. Są one krótko opisane w dalszej części tego rozdziału.

Jeżeli za pomocą jednej z opcji `F`, `f`, `I` lub `n` włączone jest przetwarzanie wsadowe, C News spodziewa się znaleźć w tym polu nazwę pliku, a nie polecenie. Jeżeli nazwa pliku nie zaczyna się od znaku ukośnika (/), zakłada się, że jest względna do */var/spool/news/out.going*. Jeżeli pole jest puste, domyślnie przyjmowana jest wartość *remote-system/togo*. Oczekuje się, że plik ma ten sam format, co plik *remote-system/togo* i zawiera listę artykułów do wysłania.

Przy konfigurowaniu C News prawdopodobnie będziesz musiał stworzyć własny plik *sys*. Oto przykładowy plik dla **vbrew.com**. Możesz z niego skopiować to, co ci jest potrzebne:

```
# Bierzemy co daj□
ME:all/all::
# Wysy□amy wszystko do moria, z wyj□tkiem artyku□ów lokalnych
# i zwi□zanych z browarem. U□ywamy przetwarzania wsadowego
moria/moria.orcnet.org:all,!to,to.moria/all,!local,!brewery:f:
# Wysy□amy comp.risks do jack@ponderosa.uucp
ponderosa:comp.risks/all::rmail jack@ponderosa.uucp
# swim otrzymuje mniej grup
swim/swim.twobirds.com:comp.os.linux,rec.humor.oracle/all,!local:f:
# Zapisujemy artyku□y mail.map do dalszego przetwarzania
usenet-maps:comp.mail.maps/all:F:/var/spool/uumaps/work/batch
```

Plik active

Plik *active* znajduje się w katalogu */etc/news* i zawiera wszystkie grupy znane twojemu ośrodkowi oraz aktualnie dostępne artykuły. Rzadko będziesz musiał z nim cokolwiek robić, ale aby opis był pełny, krótko go przedstawimy. Wpisy mają następującą postać:

grupa maks min prawa

`grupa` to nazwa grupy. *`maks`* i *`min`* to najniższy i najwyższy numer aktualnie dostępnych artykułów. Jeżeli w danej chwili żaden nie jest dostępny, *`min`* ma wartość równą *`maks+1`*. Do tego właśnie służy pole *`min`*. Jednak aby nie osłabiać działania, C News nie uaktualnia tego pola. Nie byłoby to problemem, gdyby nie istniały przeglądarki, które sięgają do tego pola. Na przykład *trn* sprawdza to pole, by zobaczyć, czy może usunąć jakieś artykuły ze swojej bazy wątków. Aby uaktualniać pole *`min`*, musisz uruchamiać regularnie polecenie *updatemin* (lub w starszych wersjach C News jego odpowiednik: skrypt *upact*).

Parametr `prawa` określa szczegółowo prawa dostępu użytkowników do danej grupy. Przyjmuje on jedną z poniższych wartości:

`y`
: Użytkownicy mają prawo wysyłać artykuły do tej grupy.

`n`
: Użytkownicy nie mają prawa wysyłać artykułów do tej grupy. Jednak wciąż mogą czytać zawarte w niej artykuły.

`x`
: Ta grupa została lokalnie zablokowana. Dzieje się tak czasem, gdy administratorzy grup (lub ich przełożeni) zezłoszczą się na pewne artykuły wysłane do jakichś grup.

  Artykuły odebrane dla tej grupy nie są zachowywane lokalnie, choć są przekazywane do ośrodków, które o nie proszą.

`m`
: Oznacza grupę moderowaną. Gdy użytkownik próbuje wysłać artykuł do tej grupy, inteligentna przeglądarka powiadamia o tym i wysyła artykuł do moderatora. Adres moderatora jest pobierany z pliku *moderators* znajdującego się w katalogu */var/lib/news*.

`=rzeczywista-grupa`
: Oznacza, że `grupa` jest lokalnym aliasem dla innej grupy o nazwie `rzeczywista-grupa`. Wszystkie artykuły wysłane do `grupy` zostaną przekierowane do grupy rzeczywistej.

W C News zwykle nie będziesz miał bezpośredniego dostępu do tego pliku. Grupy mogą być dodawane lub usuwane lokalnie za pomocą poleceń *addgroup* i *delgroup* (zobacz podrozdział *Narzędzia i zadania administracyjne* kończący ten rozdział). Wiadomość kontrolna `newgroup` dodaje grupę w całym Usenecie, a `rmgroup` ją usuwa. *Nigdy sam nie wysyłaj takiej wiadomości!* Instrukcje, jak tworzyć grupy, znajdziesz w artykułach wysyłanych co miesiąc do grupy *news.announce.newusers*.

Plik *active.times* jest ściśle związany z plikiem *active*. Gdy grupa zostanie stworzona, C News zapisuje do tego pliku komunikat zawierający nazwę utworzonej grupy, datę utworzenia, informacje, czy została utworzona przez komunikat kontrolny `newgroup`, czy lokalnie, oraz kto ją utworzył. Dane z tego pliku przydają się przeglądarkom grup, które mogą powiadamiać użytkownika o nowo utworzonych grupach. Używane są także przez polecenie *NEWGROUPS* NNTP.

Przetwarzanie wsadowe artykułów

Wsady grup dyskusyjnych są zgodne z pewnym formatem, który jest identyczny dla B News, C News i INN. Każdy artykuł jest poprzedzany następującym wierszem:

```
#! rnews liczba
```

Parametr `liczba` określa rozmiar artykułu w bajtach. Gdy używasz kompresji wsadowej, wynikowy plik jest kompresowany jako całość i poprzedzany innym wier-

szem, który informuje o tym, że plik należy rozpakować. Standardowym na rzędziem uży wa nym do kom pre sji jest *compress* i mo żna je roz po znać po na stę pującym wierszu:

```
#! cunbatch
```

Jeżeli ser wer grup wy syła wsa dy pocztą, która ze wszyst kich da nych usuwa ósmy bit, skom pres owa ny wsad należy za bezp iecz yć, używ ając tak zwa nego *kodowania-c7* (*c7-encoding*). Ta kie wsa dy są oznac zane ja ko *c7unbatch*.

Gdy wsad zo stan ie prze kaz any do *rnews* w ośrodku zdal nym, te znacz niki są sprawdzane i plik jest od pow iednio prze twar zany. Niek tóre ośrodki używ ają in nych narzędzi do kompresji, jak *gzip*, i wtedy poprzedzają skompresowane pliki słowem *zunbatch*. C News nie roz poz naje nie stand ardo wych nagłów ków jak ten. Aby były one obsługiw ane, mu sisz zmo dyf iko wać kod źródłowy.

Przetwarzanie wsadowe artykułów w C News jest realizowane za pomocą pliku */usr/lib/news/batch/sendbatches*, który bie rze li stę ar ty kułów z pli ku *site/togo* i umieszcza je w kil ku wsa dach. Po wi nien on być uru cha mia ny co go dzi nę lub na wet częściej, w za le żno ści od in ten syw no ści ru chu. Je go działanie jest kon tro lo wa ne przez plik *batchparms* znaj dujący się w ka ta lo gu */var/lib/news*. Plik ten opi su je: mak sy mal ny roz miar wsa du do pusz czal ny dla ka ż de go ośrod ka, pro gra my uży wa ne do przetwa rza nia wsa do we go i opcjo nal nej kom pre sji oraz me to dę do star cza nia pacz ki do ośrod ka zdal ne go. Pa ra me try prze twa rza nia wsa do we go możesz okre ślić od dzielnie dla ka ż de go ośrod ka. Na to miast dla ośrodków, które nie są zde fi nio wa ne nie zależnie, trze ba je okre ślić w ra mach pa rametrów do my śl nych.

Przy in sta la cji C News, naj praw do po dob niej znaj dziesz w swo jej dys try bu cji plik *batchparms* za wie rający od po wied nie wpi sy do my śl ne, a więc ist nie je du ża szan sa, że nie bę dziesz mu siał nic zmie niać w tym pli ku. Na wszel ki wy pa dek opi sze my jednak je go for mat. Ka żdy wiersz składa się z sze ściu pól od dzie lo nych spa cja mi lub tabulatorami:

```
o□rodek rozmiar maks prog_prze_wsad kompr transport
```

`o□rodek`
: `o□rodek` to nazwa ośrodka, którego dotyczy wpis. Plik *togo* dla tego ośrodka musi znajdować się w *out.goint/togo* w katalogu bufora grup. Nazwa ośrodka `/default/` oznac za do myś lny wpis i pa suje do ka żd ego ośrodka, który nie jest zdefiniowanyindywidualnymwpisem.

`rozmiar`
: `rozmiar` okreś la mak sym alny roz miar two rzon ych wsa dów ar tykułów (przed kom presją). Je żeli po jed yncze ar tykuły są wię ks ze, niż ten roz miar, C News robi wyjątek i umieszc za ka żdy z nich w od dzieln ym pli ku wsa dow ym.

`maks`
: `maks` określa maksymalną liczbę tworzonych i przygotowanych do wysłania wsad ów dla okreś lone go ośrodka. Jest przy datny w sy tua cji, gdy zdal ny ośrod ek jest przez długi czas nie czynny, gdyż za pob iega za śmie ca niu two ich kat alogów bu forowych UUCP mnó stwem wsa dów.

C News określa liczbę zakolejkowanych wsadów za pomocą skryptu *queuelen* znajdującego się w katalogu */usr/lib/news/*. Gdybyś zainstalował C News w postaci pakietu, skryptu nie trzeba by było edytować, ale gdybyś użył innego katalogu buforowego, jak na przykład UUCP Taylora, mogłaby zajść potrzeba edycji. Jeżeli nie przejmujesz się liczbą buforowanych plików (ponieważ jesteś jedyną osobą używająca komputera i nie tworzysz megabajtów artykułów), możesz zastąpić zawartość skryptu prostą dyrektywą *exit 0*.

`prog_prze_wsad`

Pole `prog_prze_wsad` zawiera polecenie używane do generowania wsadu z listy artykułów zawartej w pliku *togo*. W przypadku regularnego przesyłania, zwykle jest to *batcher*. W przypadku innych zastosowań, można użyć innych programów przetwarzania wsadowego. Na przykład protokół ihave/sendme wymaga, by lista artykułów była zamieniona na wiadomości kontrolne *ihave* lub *sendme*, które są wysyłane do grupy *to.site*. Jest to realizowane za pomocą *batchib* i *batchsm*.

`kompr`

Pole `kompr` określa polecenie realizujące kompresję. Zwykle jest to *compcun*, skrypt generujący skompresowany wsad*. Załóżmy jednak, że tworzysz skompresowany plik, używając *gzipa*, powiedzmy **`gzipcun`** (zauważ, że musisz napisać go samodzielnie). Musisz sprawdzić, czy *uncompress* w ośrodku zdalnym jest w stanie rozpoznawać pliki skompresowane programem *gzip*.

Jeżeli w ośrodku zdalnym nie ma polecenia *uncompress*, możesz wpisać *nocomp* i w ogóle nie kompresować plików.

`transport`

Ostatnie pole, `transport`, opisuje używany protokół przesyłania. Dostępne jest kilka standardowych poleceń dla różnych protokołów transportowych. Ich nazwy rozpoczynają się od *via*. Plik *sendbatches* przekazuje je do docelowego ośrodka w wierszu poleceń. Jeżeli wpis *batchparms* ma wartość różną od `/default/`, *sendbatches* pobiera nazwę ośrodka z pola `site`, obcinając wszystko po pierwszej kropce lub ukośniku włącznie. Jeżeli wpis `batchparms` ma wartość `/default/`, używane są nazwy katalogów z pliku *out.going*.

Aby zrealizować przetwarzanie wsadowe dla zadanego ośrodka, użyj poniższego polecenia:

```
# su news -c "/usr/lib/news/batch/sendbatches site"
```

sendbatches wywoływane bez argumentów obsługuje wszystkie zakolejkowane wsady. Interpretacja „all" zależy od obecności domyślnego wpisu w *batchparms*. Jeżeli zostanie on znaleziony, sprawdzane są wszystkie podkatalogi */var/spool/news/out.going*. W przeciwnym razie *sendbatches* wykorzystuje wszystkie kolejne wpisy w *batchparms*, obsługując znalezione tam ośrodki. Zwróć uwagę, że *sendbatches* przy

* Wraz z C News jest rozpowszechniany *compcun* wykorzystujący *compress* z opcją 12-bitową, ponieważ jest to najmniejszy wspólny mianownik dla większości ośrodków. Możesz stworzyć skrypt, powiedzmy *compcun16*, który będzie używał kompresji 16-bitowej. Jednak poprawa nie jest znacząca.

przeglądaniu katalogu *out.going* uwzględnia tylko te katalogi, które nie zawierają kropek ani znaków @ w nazwach ośrodków.

Istnieją dwa polecenia używające *uux* do wywołania *rnews* w ośrodku zdalnym: *viauux* i *viauuxz*. To ostatnie ustawia znacznik *–z* dla *uux*, by starsze wersje nie zwracały informacji o poprawnym dostarczeniu każdego artykułu. Inne polecenie, *viamail*, wysyła wsady artykułów pocztą do użytkownika **rnews** w systemie zdalnym. Oczywiście wymaga to, by system zdalny jakoś dostarczał wszystkie poczty przeznaczone dla **rnews** do swojego lokalnego systemu grup dyskusyjnych. Pełną listę protokołów transportowych znajdziesz na stronie podręcznika elektronicznego *newsbatch*.

Wszystkie polecenia z ostatnich trzech pól muszą być umieszczone w katalogu *out.going/site* lub */usr/lib/news/batch*. Większość z nich to skrypty. Możesz łatwo dołączać nowe, potrzebne ci narzędzia. Są one wywoływane przez potoki. Lista artykułów jest dostarczana programowi przetwarzania wsadowego na jego standardowe wejście, natomiast wsad dostajemy na jego standardowym wyjściu. Dalej jest on przekazywany przez potok do programu kompresującego go i tak dalej.

Oto przykładowy plik:

```
# plik batchparms dla browaru
# o□rodek       | rozmiar |maks| prog_prze_wsad | kompr | trans
#--------------+---------+----+----------------+-------+------
/default/        100000    22    batcher         compcun viauux
swim             10000     10    batcher         nocomp  viauux
```

Wygasanie grup dyskusyjnych

W B News wygaszanie musi być realizowane przez program *expire*, który jako argumenty przyjmuje listę grup wraz z czasem, po którym wygasają artykuły. Aby różne hierarchie mogły wygasać po różnym czasie, musisz napisać skrypt, który będzie wywoływał *expire* dla każdej z nich niezależnie. C News oferuje wygodniejsze rozwiązanie. W pliku *explist* możesz określić grupy i czasy ich wygaśnięcia. Polecenie *doexpire* zwykle jest wywoływane raz dziennie z *crona* i przetwarza wszystkie grupy zgodnie z listą.

Czasem będziesz chciał dłużej zatrzymać artykuły z pewnych grup, na przykład programy wysłane do grupy *comp.sources.unix*. Nazywa się to *archiwizacją*. W *explist* można wskazać grupy, które chcesz archiwizować.

Wpis w *explist* wygląda tak:

```
listagrup prawa czas archiwum
```

`listagrup` to oddzielana przecinkami lista grup dyskusyjnych, których dotyczy wpis. Hierarchie mogą być określane przez podanie przedrostka nazwy grupy z opcjonalnym słowom `all`. Na przykład w przypadku wpisu dotyczącego wszystkich grup *comp.os*, wprowadź `comp.os` lub `comp.os.all`.

Przy *wygasaniu* artykułów w grupie, nazwa jest sprawdzana we wszystkich wpisach w pliku *explist* w podanej kolejności. Wykorzystywany jest pierwszy pasujący wpis.

Na przykład, aby wyrzucić po czterech dniach większość artykułów z grup *comp*, z wyjątkiem grupy *comp.os.linux.announce*, którą chcesz przechować przez tydzień, po prostu musisz mieć dla tej ostatniej grupy wpis określający, że wygasa ona po siedmiu dniach, a dalej wpis dotyczący okresu wygaśnięcia *comp* po czterech dniach.

Pole `prawa` zawiera szczegóły, czy wpis dotyczy grup moderowanych, niemoderowanych, czy wszystkich. Może przyjmować wartości `m`, `u` lub `x`, które oznaczają odpowiednio grupy moderowane, niemoderowane lub dowolne.

Trzecie pole, `czas`, zwykle zawiera tylko jedną liczbę, która wskazuje, po ilu dniach artykuły wygasają, o ile w nagłówku artykułu nie ma pola `Expires:` określającego inną datę. Zauważ, że jest to liczba dni liczona od dnia *dotarcia* artykułu do twojego ośrodka, a nie od daty wysłania artykułu do grupy.

Pole `czas` może jednak być bardziej złożone. Są to trzy liczby oddzielone od siebie myślnikami. Pierwszy segment określa wtedy liczbę dni, która musi minąć, zanim artykuł zostanie uznany za kandydata do wygaśnięcia, nawet jeżeli pole `Expires:` już wygasło. Używanie tu innej wartości niż zero zwykle nie ma sensu. Drugi segment to poprzednio wspomniana domyślna liczba dni, po których wygasa czas przechowywania artykułu. Trzeci segment to liczba dni, po której czas dla artykułu wygasa bezwarunkowo, bez względu na to, czy zawiera pole `Expires:`, czy też nie. Jeżeli zostanie podany tylko środkowy segment, pozostałe dwa przyjmują wartości domyślne. Mogą one być zdefiniowane przez specjalny wpis `/bounds/`, który opiszemy nieco dalej.

Czwarte pole, `archiwum`, określa, czy grupa dyskusyjna ma być archiwizowana i gdzie. Jeżeli nie zamierzamy jej archiwizować, powinniśmy użyć myślnika. W przeciwnym razie użyj pełnej ścieżki (wskazującej katalog) lub znaku @. Znak @ wskazuje domyślny katalog archiwum, który musi być następnie podany w wierszu poleceń *doexpire* za pomocą znacznika *-a*. Katalog archiwum powinien być własnością użytkownika **news**. Gdy *doexpire* archiwizuje artykuły, powiedzmy z grupy *comp.sources.unix*, zachowuje je w podkatalogu *comp/sources/unix* katalogu archiwum, tworząc je, jeżeli zajdzie potrzeba. Sam katalog archiwum nie zostanie jednak stworzony.

W pliku *explist* znajdują się dwa specjalne wpisy, na których opiera się *doexpire*. Zamiast listy grup dyskusyjnych zawierają one słowa kluczowe `/bounds/` i `/expired/`. Wpis `/bounds/` zawiera domyślne wartości dla trzech segmentów opisanego powyżej pola `czas`.

Pole `/expired/` określa, jak długo C News przechowuje wiersze w pliku *history*. C News nie usuwa wiersza z pliku historii zaraz po wygaśnięciu odpowiadającego mu artykułu, ale przechowuje go na wypadek, gdyby przyszedł jego duplikat. Jeżeli grupy dostajesz tylko z jednego ośrodka, ta wartość może być niewielka. W przeciwnym razie zaleca się ustawić okres kilkutygodniowy w sieciach UUCP w zależności od doświadczenia w opóźnieniach artykułów przychodzących z różnych ośrodków.

Oto przykładowy plik *explist* o raczej krótkich okresach wygaśnięcia:

```
# przechowywanie historii przez dwa tygodnie. □aden artyku□
# nie b□dzie przechowywany d□u□ej ni□ trzy miesi□ce
/expired/                x   14      -
/bounds/                 x   0-1-90  -
# grupy, które chcemy przechowywa□ d□u□ej ni□ reszt□
comp.os.linux.announce   m   10      -
comp.os.linux            x   5       -
alt.folklore.computers   u   10      -
rec.humor.oracle         m   10      -
soc.feminism             m   10      -
# Archiwum grup *.sources
comp.sources,alt.sources x   5       @
# domy□lne warto□ci dla grup technicznych
comp,sci                 x   7       -
# wystarczaj□co na d□ugi weekend
misc,talk                x   4       -
# szybkie usuwanie □mieci
junk                     x   1       -
# oraz niezbyt ciekawych wiadomo□ci kontrolnych
control                  x   1       -
# i wpis dla pozosta□ych rzeczy
all                      x   2       -
```

Wygasanie stwarza kilka potencjalnych problemów. Jednym z nich jest to, że twoja przeglądarka grup może opierać się na trzecim polu pliku *active* opisanym wcześniej, zawierającym najmniejszy numer aktualnie dostępnego artykułu. Gdy artykuły wygasają, C News nie uaktualnia tego pola. Jeżeli potrzebujesz (lub chcesz), by pole to odzwierciedlało rzeczywistą sytuację, musisz uruchomić program *updatemin* po każdym uruchomieniu *doexpire*. (W starszych wersjach C News robi to skrypt *upact*).

C News nie realizuje wygasania przez przeglądanie katalogów grup, a po prostu sprawdza w pliku *history*, czy czas przechowywania artykułu ma wygasnąć*. Jeżeli plik historii w jakiś sposób się rozsynchronizuje, artykuły mogą pozostać na dysku na zawsze, ponieważ C News o nich zapomni**. Możesz to naprawić, używając skryptu *addmissing* znajdującego się w katalogu */usr/lib/news/maint*, który doda brakujące artykuły do pliku *history* lub *mkhistory*, który przebuduje cały plik od początku. Nie zapomnij przed wywołaniem tych poleceń wejść na konto użytkownika **news**, gdyż w przeciwnym razie plik *history* będzie nieczytelny dla C News.

Różne dodatkowe pliki

Istnieje szereg plików, które kontrolują zachowanie C News, ale nie są istotne. Wszystkie znajdują się w katalogu */etc/news*. Krótko je tutaj opiszemy:

---

* Data przyjścia artykułu jest zawarta w środkowym polu wiersza historii i jest zapisana jako liczba sekund od 1 stycznia 1970 roku.

** Nie wiem *dlaczego*, ale od czasu do czasu to się zdarza.

newsgroups

Jest to plik towarzyszący plikowi *active*, zawierający listę wszystkich grup dyskusyjnych wraz z jednowierszowym opisem. Plik ten jest automatycznie uaktualniany, gdy C News odbierze wiadomość kontrolną `checknews`.

localgroups

Jeżeli posiadasz wiele grup lokalnych, C News będzie informować o nich za każdym razem, gdy dostaniesz wiadomość `checkgroups`. Można temu zapobiec, umieszczając nazwy ich grup i opisy w pliku w formacie takim jak *newsgroups*.

mailpaths

Ten plik zawiera adres moderatora dla każdej grupy moderowanej. Każdy wiersz zawiera nazwę grupy, a po niej adres e-mail moderatora (oddzielone tabulatorem).

Domyślnie dodawane są dwa specjalne wpisy: `backbone` i `internet`. Oba są zapisane w notacji wykazu trasowania i zawierają odpowiednio ścieżkę do najbliższego ośrodka szkieletowego oraz ośrodka, który rozumie adresy RFC-822 (*użytkownik@host*). Domyślne wpisy są następujące:

```
internet      backbone
```

Nie musisz zmieniać wpisu `internet`, jeżeli masz zainstalowanego Exima lub *sendmail*. Rozumieją one adresowanie RFC-822.

Wpis `backbone` stosuje się wtedy, gdy użytkownik wysyła artykuł do grupy moderowanej, której moderator nie jest wpisany bezpośrednio. Jeżeli nazwa grupy to *alt.sewer*, a `backbone` zawiera wpis *path!%s*, C News wyśle artykuł pocztą e-mail na adres *path!alt-sewer*, mając nadzieję, że maszyna szkieletowa będzie w stanie przekazać go dalej. Możesz zapytać administratora grup na serwerze, od którego je dostajesz, jakiej ścieżki masz użyć. W ostateczności możesz użyć także *uunet.uu.net!%s*.

distributions

Ten plik w rzeczywistości nie jest plikiem C News, ale jest używany przez niektóre przeglądarki grup i *nntpd*. Zawiera listę dystrybucji rozpoznawanych przez twój ośrodek i opis ich (zamierzonego) działania. Na przykład browar wirtualny posiada następujący plik:

```
world       Everywhere in the world
local       Only local to this site
nl          Netherlands only
mugnet      MUGNET only
fr          France only
de          Germany only
brewery     Virtual Brewery only
```

log

Ten plik zawiera zapis wszystkich działań C News. Jest on regularnie czyszczony przez *newsdaily*. Kopie starych plików log są przechowywane w *log.o*, *log.oo* itp.

errlog

Jest to za pis wszyst kich komuni katów błędów wy stęp ujących w C News. Nie zawier ają one za pisów na te mat ar tykułów prze nies iony ch do śmieci ze wzglę du na błędną gru pę lub inne błędy. Ten plik, o ile nie jest pu sty, jest au tom aty cznie wysyłany pocztą e-ma il do zarządcy grup (do myś lnie do użytk owni ka **usenet**) przez pro gram *newsdaily*.

errlog jest czyszczony przez newsdaily. *errlog.o* przechowuje ko pie sta rych plików i tym po dob ne.

batchlog

Ten plik za wiera za pis wszyst kich uruc homi eń *sendbatches* i naj częś ciej jest mało cie kawy. Zwy kle jest też obsługiw any przez *newsda ily*.

watchtime

Jest to pu sty plik two rzony przy ka żdym uruc homi eniu *newsdaily*.

Wiadomości kontrolne

Pro tokół grup usenetowych ro zu mie spe cjalną ka te go rię ar ty kułów, kt óre wy wołują pewne odpowiedzi lub działania w systemie grup dyskusyjnych. Są to tak zwa-ne wia do mo ści *kontrolne*. Ich cechą cha rak te ry styczną jest obec ność po la `Control:` w nagłów ku ar ty kułu. Za wie ra ono na zwę działania do wy ko na nia. Ist nie je kil ka typów działań, a wszystkie są obsługiwane przez skrypty powłoki umieszczone w ka ta lo gu */usr/lib/news/ctl*.

Wię k szość z tych wia do mo ści wy ko nu je swo je za da nia au to ma tycz nie w mo mencie prze twa rza nia ar ty kułu przez C News i bez po wia da mia nia zarządcy grup. Domy śl nie tyl ko wia do mość `checkgroups` bę dzie obsługi wa na przez zarządcę, ale możesz to zmie nić edy tując skryp ty.

cancel

Naj bar dziej znaną wia do mo ścią jest `cancel`, dzię ki któr ej użyt kow nik może anu lować wcze śniej wysłany ar ty kuł. Usu wa ona sku tecz nie ar ty kuł z ka ta logów bu fo ra, je żeli tam ist nie je. Wia do mość `cancel` jest prze ka zy wa na do wszyst kich ośrodk ów, które ode brały wia do mość w da nej gru pie, bez wzglę du na to, czy ar ty kuł był już czy ta ny. Ist nie je ry zy ko, że wia do mość ta przyj dzie wcze śniej, niż ar ty kuł do anu lowania . Nie kt óre sys te my grup po zwa lają użyt kow ni kom anu lo wać wia do mo ści innych os ób.

newgro up i rmgro up

Dwie wia do mo ści służące do two rze nia i usu wa nia grup to `newgroup` i `rmgroup`. Grupy w „zwykłych" hierarchiach mogą być tworzone jedynie po uzgodnieniu i głosowaniu przeprowadzonym wśród czytelników Usenetu. Reguły dotyczące hie rar chii *alt* po zwa lają na coś zbliż one go do anar chii. Wię cej in for ma cji na ten te mat znajdziesz w artykułach gru py *news.announce.newusers* i *news.announce.newgroups*.

Nigdy nie wysyłaj samodzielnie wiadomości `newgroup` i `rmgroup`, jeżeli nie jesteś pewny, czy masz do tego prawo.

checkgroups

Wiadomości `checkgroups` są wysyłane przez administratora grup w celu synchronizacji plików *active* we wszystkich ośrodkach w sieci Usenet. Na przykład komercyjny dostawca Internetu może wysłać taką wiadomość do ośrodków swoich klientów. Raz w miesiącu przez moderatora grupy *comp.announce.newgroups* jest wysyłana „oficjalna" wiadomość `checkgroups` dla głównych hierarchii. Jednak jest ona wysyłana jako zwykły artykuł, a nie wiadomość kontrolna. Aby wykonać operację `checkgroups`, zapisz artykuł do pliku, powiedzmy */tmp/check*, usuń cały początek samej wiadomości kontrolnej i przekaż ją do skryptu `checkgroups` za pomocą następującego polecenia:

```
# su news -c "/usr/lib/news/ctl/checkgroups" < /tmp/check
```

Twój plik *newsgroups* zostanie uaktualniony na podstawie nowej listy grup. Dodane zostaną grupy wymienione w pliku *localgroups*. Stary plik *newsgroups* zostanie przemianowany na *newsgroups.bac*. Zauważ, że wysłanie wiadomości lokalnie rzadko działa, ponieważ *inews*, polecenie przyjmujące i wysyłające artykuły od użytkowników, odmawia przyjęcia tak dużego artykułu.

Gdyby C News stwierdził różnice pomiędzy `checkgroups` a plikiem *active*, wygenerowałby listę poleceń, które uaktualniłyby twój ośrodek, i wysłałby ją do administratora grup dyskusyjnych.

Wynik zwykle wygląda jakoś tak:

```
From news Sun Jan 30 16:18:11 1994
Date: Sun, 30 Jan 94 16:18 MET
From: news (News Subsystem)
To: usenet
Subject: Problems with your active file
The following newsgroups are not valid and should be removed.
   alt.ascii-art
   bionet.molbio.gene-org
   comp.windows.x.intriscis
   de.answers
You can do this by executing the commands:
   /usr/lib/news/maint/delgroup alt.ascii.art
   /usr/lib/news/maint/delgroup bionet.molbio.gene-org
   /usr/lib/news/maint/delgroup comp.windows.x.intrisics
   /usr/lib/news/maint/delgroup de.answers
The following newsgroups were missing.
   comp.binaries.cbm
   comp.databases.rdb
   comp.os.geos
   comp.os.gnx
   comp.unix.user-friendly
   misc.legal.moderated
   news.newsites
   soc.culture.scientists
   talk.politics.crypto
   talk.politics.tibet
```

Gdy odbierzesz tego typu wiadomość od swojego systemu grup, nie ufaj jej bezkrytycznie. W zależności od tego, kto wysłał ci wiadomość `checkgroups`, może brakować kilku grup lub nawet całych hierarchii. Powinieneś uważać przy usuwaniu jakichkolwiek grup. Jeżeli dostaniesz informację, że brakuje jakichś grup, które powinieneś mieć u siebie, musisz dodać je za pomocą skryptu *addgroup*. Zachowaj tę listę brakujących grup w pliku i przekaż do poniższego skryptu:

```
#!/bin/sh
#
WHOIAM='whoami'
if [ "$WHOIAM" != "news" ]
then
   echo "You must run $0 as user 'news'" >&2
   exit 1
fi
#
cd /usr/lib/news
while read group; do
   if grep -si "^$group[[:space:]].*moderated" newsgroup; then
      mod=m
   else
      mod=y
   fi
   /usr/lib/news/maint/addgroup $group $mod
done
```

sendsys, version i senduuname

Istnieją trzy wiadomości, których można użyć do poznania topologii sieci. Są to: `sendsys`, `version` i `senduuname`. Powodują one, że C News zwraca do nadawcy odpowiednio: plik *sys*, ciąg znaków zawierający wersję oprogramowania oraz wynik polecenia *uuname*. C News bardzo lakonicznie podchodzi do wiadomości `version`, gdyż zwraca tylko `C`.

Nie powinieneś używać tych wiadomości, jeśli nie masz pewności, że nie wyjdą poza twoją (regionalną) sieć. Odpowiedzi na wiadomość `sendsys` mogą łatwo uszkodzić sieć UUCP*.

C News w środowisku NFS

Prostym sposobem na rozpowszechnianie wiadomości w sieci lokalnej jest trzymanie wszystkich grup na centralnym hoście i eksportowanie istotnych katalogów przez NFS, tak by przeglądarki mogły skanować artykuły bezpośrednio. Nadmiarowość wymagana do odbierania i podziału artykułów na wątki jest znacznie niższa niż przy protokole NNTP. Z drugiej strony NNTP wygrywa w sieciach heterogenicznych, gdzie hosty znacznie różnią się sprzętowo lub gdzie użytkownicy nie mają identycznych kont na maszynie serwera.

Gdy używasz NFS-a, artykuły wysłane do hosta lokalnego muszą być przekazane do komputera centralnego. Inaczej pliki są narażone na niespójność, ponieważ zaw-

* Nie próbowałbym tego w Internecie.

sze istnieje ryzyko powstania wyścigów. Możesz także zabezpieczyć obszar bufora grup dyskusyjnych, eksportując go tylko do odczytu, co także wymaga przekazywania do komputera centralnego.

C News obsługuje taką konfigurację z komputerem centralnym w sposób przezroczysty dla użytkownika. Gdy wysyłasz artykuł, twoja przeglądarka grup zwykle wywołuje *inews*, by wrzucić artykuł do systemu grup. To polecenie sprawdza artykuł, uzupełnia nagłówek i sprawdza plik *server* w katalogu */etc/news*. Jeżeli plik istnieje i zawiera nazwę hosta inną niż nazwa hosta lokalnego, *inews* jest wywoływany na tym hoście przez *rsh*. Ponieważ skrypt *inews* używa kilku poleceń i obsługuje pliki C News, musisz mieć lokalnie zainstalowane C News lub zamontowane oprogramowanie z serwera.

Aby wywołanie *rsh* działało poprawnie, każdy użytkownik, który wysyła wiadomości, musi mieć takie samo konto na serwerze, to znaczy takie, na które może się zalogować bez hasła.

Sprawdź, czy nazwa hosta wpisana w pliku *server* jest identyczna z wynikiem polecenia *hostname* na serwerze. Jeśli nie – C News będzie w nieskończoność próbował dostarczyć artykuł. NFS omawiamy szczegółowo w rozdziale 14, *Sieciowy system plików*.

Narzędzia i zadania administracyjne

Pomimo złożoności C News, życie administratora grup może być całkiem przyjemne. C News posiada bowiem szereg narzędzi administracyjnych. Niektóre z nich są pomyślane do regularnego uruchamiania z *crona*, podobnie jak *newsdaily*. Skrypty te wyręczają cię w wielu codziennych zadaniach administracyjnych.

Jeżeli nie powiedziano inaczej, te polecenia administracyjne znajdują się w katalogu */usr/lib/news/maint**:

newsdaily
: Nazwa mówi sama za siebie: uruchom raz dziennie. Jest to ważny skrypt, który pomaga ci utrzymać małe rozmiary plików log, pozostawiając kopię każdego z nich z trzech ostatnich przebiegów. Próbuje także wykryć nieprawidłowości takie, jak stare wsady w katalogach przychodzących i wychodzących, wysyłanie do nieznanych lub moderowanych grup itp. Zwracane komunikaty błędów są wysyłane do administratora grup.

newswatch
: Ten skrypt powinien być uruchamiany regularnie, co godzinę, w celu wyszukiwania nieprawidłowości w systemie grup. Służy on do wykrywania problemów, które mają natychmiastowy wpływ na działanie twojego systemu grup. Sprawdza stare pliki blokujące, które nie zostały usunięte, nie obsłużone wsady i miejsce na dysku twardym. Jeśli *newswatch* wykryje problem, wysyła informację do administratora grup.

---

* Zauważ, że musisz być użytkownikiem **news**, zanim wywołasz te polecenia. Uruchomienie ich z powłoki superużytkownika może spowodować, że krytyczne pliki staną się niedostępne dla C News.

addgroup

Ten skrypt dodaje lokalnie grupę do twojego ośrodka. Poprawne wywołanie to:

```
addgroup nazwagrupy y|n|m=rzeczywistagrupa
```

Drugi argument odpowiada znacznikowi w pliku *active*, czyli każdy może wysyłać do grupy (`y`), nikt nie może wysyłać (`n`) i jest to grupa moderowana (`m`) lub że jest to alias do innej grupy (`=rzeczywistagrupa`). Możesz także używać *addgroup*, gdy pierwszy artykuł do nowo utworzonej grupy przyjdzie wcześniej niż wiadomość kontrol na newgroup, która ma zadanie tę grupę utworzyć.

delgroup

Ten skrypt pozwala na usunięcie lokalne grupy. Wywołanie jest następujące:

```
delgroup nazwagrupy
```

Nieustannie musisz usuwać artykuły, które pozostają w katalogu buforowym grupy. Ewentualnie możesz pozostawić je naturalnej kolei rzeczy (to znaczy do wygaśnięcia czasu ich przechowywania).

addmissing

Ten skrypt dodaje brakujące artykuły do pliku *history*. Uruchom go, gdy istnieją artykuły, które wydają się zalegać od zawsze.

newsboot

Ten skrypt powinien być uruchamiany w czasie inicjacji systemu. Usuwa wszelkie pliki blokujące pozostałe przy unicestwianiu procesów i zamyka oraz uruchamia wszelkie pozostałe wsady z połączeń NNTP, które zostały przerwane przez zamknięcie systemu.

newsrunning

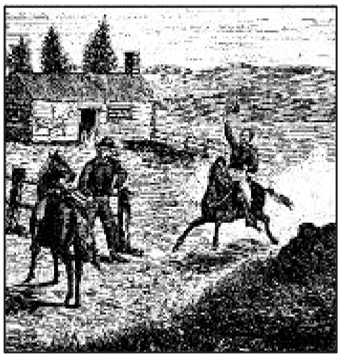
Ten skrypt znajduje się w katalogu */usr/lib/news/input* i może być użyty do zablokowania rozpakowywania wsadów przychodzących wiadomości, na przykład w czasie godzin pracy. Możesz wyłączyć rozpakowywanie wsadów wywołując:

```
/usr/lib/news/input/newsrunning off
```

Włącza się je, używając `on` zamiast `off`.

22

NNTP i demon nntpd



Protokół przesyłania wiadomości w sieci Usenet, NNTP (*Network News Transfer Protocol*), reprezentuje zupełnie odmienne podejście do wymiany grup dyskusyjnych, niż C News i inne serwery grup bez wbudowanej obsługi NNTP. Do przesyłania artykułów pomiędzy maszynami nie korzysta z technologii wsadowej charakterystycznej dla UUCP, ale pozwala wymieniać artykuły przez interaktywne połączenie sieciowe. NNTP nie jest pakietem oprogramowania, ale standardem internetowym opisanym w RFC-977. Korzysta z połączeń strumieniowych, zwykle działających w oparciu o TCP pomiędzy klientem w sieci a serwerem, który przechowuje grupy na swoim dysku lokalnym. Połączenie strumieniowe pozwala klientowi i serwerowi na interaktywne negocjowanie przesyłania artykułów prawie bez opóźnień, za czym idzie mały stopień ich dublowania. Jeśli uwzględnimy jeszcze wysoką przepustowość Internetu, otrzymujemy rozwiązanie znacznie przewyższające możliwości dotychczasowego UUCP. Choć jeszcze kilka lat temu nie było niczym niezwykłym, że artykuł szedł dwa tygodnie lub dłużej, zanim dotarł na drugi koniec sieci Usenet, to teraz trwa to zwykle krócej niż dwa dni. W samym Internecie są to nawet minuty.

Różne polecenia pozwalają klientom odbierać, wysyłać i umieszczać w grupie artykuły. Różnica pomiędzy wysyłaniem a umieszczaniem w grupie polega na tym, że umieszczanie dotyczy artykułów, które mogą mieć niepełne informacje w nagłówku. Ogólnie oznacza to, że użytkownik po prostu napisał artykuł*. Artykuły mogą być pobierane zarówno przez klientów przesyłających wiadomości, jak i przez przeglądarki grup dyskusyjnych. Dlatego NNTP jest idealnym narzędziem, które daje dostęp do grup wielu klientom w sieci lokalnej, bez gimnastyki cechującej korzystanie z NFS-a.

NNTP zapewnia także czynny i bierny sposób przesyłania grup, potocznie zwany „wciskaniem” i „ściąganiem”. Wciskanie w zasadzie przypomina protokół ihave/sendme używany przez C News (opisany w rozdziale 21, *C News*). Klient oferuje arty-

---

* Przy umieszczaniu artykułu przez NNTP, serwer zawsze dodaje przynajmniej jedno pole nagłówka `NNTP-Posting-Host:`. Pole to zawiera nazwę hosta klienta.

kuł ser we ro wi po przez po le ce nie *IHAVE msgid*, a ser wer zwra ca w od po wie dzi kod, któ ry mó wi, czy ma już ten ar ty kuł lub czy też go chce. Je żeli ser wer chce ar ty kuł, klient wy syła go, ko ńcząc tekst wier szem za wie rającym je dy nie krop kę.

Wci skan ie wia dom ości ma jedną wa dę – ob ciąża ser wer – po niew aż sys tem mu si prze szuk iwać ba zę hi stor ii dla ka żd ego po jed ynczego ar tykułu.

Dru ga tech ni ka, ściąga nie wia do mo ści, po le ga na tym, że klient pro si o li stę wszyst kich (do stęp nych) ar ty kułów z grup, któ re do tarły w ja ki mś dniu. To za py ta nie jest re ali zo wa ne przez po le ce nie *NEWNEWS*. Ze zwró co nej li sty ID wia do mo ści klient wy bie ra te nu me ry, których mu jesz cze bra ku je, wy dając dla ka ż de go z nich po le ce nie *ARTICLE*.

Ściąga nie grup wy ma ga od ser we ra ścisłego kon tro lo wa nia, które gru py i dys try bu cje pozwa la ściągać klien to wi. Na przykład mu si za gwa ran to wać, że żad ne taj ne ma te riały z grup lo kal nych dla da ne go ośrod ka nie zo staną wysłane do nie au to ry zo wa nych kli en tów.

Istnieje też kilka poleceń wygodnych dla przeglądarek grup. Za ich pomocą może od bie rać od dziel nie nagłówek i tre ść ar ty kułu lub na wet po je dyn cze wier sze nagłówka z zadanego zakresu artykułów. Pozwala to trzymać wszystkie grupy na ho ście cen tral nym i mieć użyt kow ni ków w sie ci (przy pusz czal nie lo kal nej), któ rzy za po mocą klien ta NNTP czy tają je i wy syłają. Jest to roz wiąza nie al ter na tyw ne do eks por to wa nia ka ta lo gów z gru pa mi przez NFS, co zo stało opi sa ne w roz dzia le 21.

Man ka men tem NNTP jest to, że znającej się na rze czy oso bie pro to kół ten umo żli wia wsta wie nie w stru mień grup ar ty kułu z fałszywą in for ma cją o nadaw cy. Na zy wa się to *fałszowaniem* (ang. *news faking*) lub *podszywaniem* (ang. *spoofing*)*. Rozszerzenie NNTP po zwa la na uwie rzy tel nia nie użyt kow ni ków przy pew nych po le ce niach, co ja koś za bez pie cza przed nad uży wa niem two je go ser we ra grup dys ku syj nych.

Ist nie je sze reg pa kie tów NNTP. Jed nym z bar dziej po pu lar nych jest de mon NNTP, zna ny ta kże ja ko *implemen tacja wzorcowa* (ang. *reference implemen tation*). Zo stał na pi sa ny przez Sta na Bar be ra i Phi la Lap sleya ja ko ilu stra cja RFC-977. Po dob nie jak wię k szość do bre go opro gra mo wa nia, tak i ten pa kiet możesz obec nie zna le źć w swo jej dys try bu cji Linuk sa. Możesz też po brać je go kod źródłowy i skom pi lo wać sa mo dziel nie pod wa run kiem, że na ty le do brze znasz swoją dys try bu cję Linuk sa, by po praw nie skon fi gu ro wać wszel kie ście żki do pli ków.

Pa kiet *nntpd* za wiera ser wer, dwa klien ty ściągające i wci skające wia dom ości oraz za mienn ik dla *inews*. Działają one w środ owi sku B News, ale po nie wielk ich zmia nach będą ta kże działać z C News. Jed nak, je żeli pla nuj esz używ ać NNTP nie tyl ko do udos tępni ania grup dys kus yjny ch na two im ser wer ze, im plem enta cja wzor cowa nie jest do brym wy bor em. Dla tego omó wimy tyl ko de mona NNTP za wart ego w pa kiec ie *nntpd*, a pro gramy klienc kie po zos tawi my w spo koju.

* Ten sam pro blem wy stę pu je w pro to ko le SMTP, choć obec nie wię k szość age n tów trans por towych pocz ty posia da mecha nizm za po bie gający pod szy wa niu.

Gdybyś chciał uruchomić duży ośrodek grup dyskusyjnych, powinieneś zainteresować się pakietem *InterNetNews*, inaczej INN, napisanym przez Richa Salza. Zapewnia on transport grup zarówno przez NNTP, jak i UUCP, co jest zdecydowanie lepsze niż *nntpd*. INN omawiamy szczegółowo w rozdziale 23, *InternetNews*.

Protokół NNTP

Wspomnieliśmy o dwóch poleceniach, które decydują o tym, jak artykuły są wciskane lub ściągane pomiędzy serwerami. Teraz przyjrzymy się im w kontekście rzeczywistej sesji NNTP, a przekonasz się, jak prosty jest ten protokół. Użyjemy prostego klienta *telnet*, za pomocą którego podłączymy się do serwera opartego na INN, działającego w browarze wirtualnym pod adresem **news.vbrew.com**. Żeby nie wydłużać niepotrzebnie przykładu, serwer działa w minimalnej konfiguracji. Pełną konfigurację tego serwera poznamy w rozdziale 23. W naszych testach będziemy bardzo ostrożni i wygenerujemy artykuły do grupy *junk*, żeby nie zakłócać innym spokoju.

Podłączanie się do serwera grup

Podłączanie się do serwera polega na otwarciu połączenia TCP do jego portu NNTP. Gdy jesteś podłączony, pojawi się baner powitalny. Jednym z pierwszych poleceń, jakie możesz wypróbować jest `help`. Odpowiedź na nie przeważnie zależy od tego, czy serwer widzi cię jako zdalny serwer NNTP, czy jako przeglądarkę grup. Udostępnia wtedy różne zestawy poleceń. Swój tryb działania możesz zmienić, wydając polecenie *mode*. Przyjrzymy się mu za chwilę.

```
$ telnet news.vbrew.com nntp
Trying 172.16.1.1...
Connected to localhost.
Escape character is '^]'.
200 news.vbrew.com InterNetNews server INN 1.7.2 08-Dec-1997 ready
help
100 Legal commands
   authinfo
      help
      ihave
      check
      takethis
      list
      mode
      xmode
      quit
      head
      stat
      xbatch
      xpath
      xreplic
For more information, contact "usenet" at this machine.
.
```

Odpowiedzi na polecenia NNTP zawsze kończą się kropką (.) w oddzielnym wierszu. Liczby, które widzisz w wyniku, to *kody odpowiedzi* używane przez serwer do

wskazania, czy polecenie zostało wykonane poprawnie, czy błędnie. Kody odpowiedzi są opisane w RFC-977. Najważniejsze z nich omówimy dalej.

Wciskanie artykułu do serwera

Przy omawianiu wciskania artykułów do serwera, wspomnieliśmy o poleceniu *IHAVE*. Przyjrzyjmy się teraz, jak w rzeczywistości działa to polecenie:

```
ihave <123456@gw.vk2ktj.ampr.org>
335
From: terry@gw.vk2ktj.ampr.org
Subject: test message sent with ihave
Newsgroups: junk
Distribution: world
Path: gw.vk2ktj.ampr.org
Date: 26 April 1999
Message-ID: <123456@gw.vk2ktj.ampr.org>
Body:

This is a test message sent using the NNTP IHAVE command.

.
235
```

We wszystkich poleceniach NNTP nie istotna jest pisownia, a więc możesz używać zarówno małych, jak i dużych liter. Polecenie *IHAVE* przyjmuje jeden obowiązkowy argument – ID wiadomości, która jest wciskana. Każdemu artykułowi w czasie jego tworzenia jest przypisywany unikatowy numer ID. Polecenie *IHAVE* stanowi sposób na powiedzenie przez serwer NNTP, które artykuły posiada i które chce wrzucić do innego serwera. Serwer wysyłający wydaje polecenie *IHAVE* dla każdego artykułu, który chce wrzucić. Jeżeli kod odpowiedzi na polecenie wygenerowany przez odbierający serwer NNTP jest z zakresu „3xx", wysyłający serwer NNTP prześle pełny artykuł, włącznie z jego nagłówkiem, zakończy go kropką w oddzielnym wierszu. Jeżeli kod odpowiedzi należy do zakresu „4xx", serwer odbierający nie przyjmie tego artykułu, prawdopodobnie dlatego, że go ma lub z innego powodu, na przykład mogło mu zabraknąć miejsca na dysku.

Jeśli artykuł został przesłany, serwer odbierający zwraca inny kod odpowiedzi, mówiący, czy przesłanie artykułu zakończyło się poprawnie.

Przejście do trybu czytania NNRP

Przeglądarki grup używają do komunikacji z serwerem własnego zestawu poleceń. Aby je uaktywnić, serwer musi być w trybie *czytania*. Większość serwerów grup dyskusyjnych domyślnie jest w trybie czytania, chyba że adres IP podłączające go się hosta znajduje się na liście partnerów do przekazywania grup. W każdym razie NNTP posiada polecenie jawnie przełączające serwer do trybu czytania:

```
mode reader
200 news.vbrew.com InterNetNews NNRP server INN 1.7.2 08-Dec-1997 ready/(posting ok).
help
100 Legal commands
   authinfo user Name|pass Password|generic <prog> <args>
   article [MessageID|Number]
```

```
   body [MessageID|Number]
   date
   group newsgroup
   head [MessageID|Number]
   help
   ihave
   last
   list [active|active.times|newsgroups|distributions|distrib.pats|/
        overview.fmt|subscriptions]
   listgroup newsgroup
   mode reader
   newgroups yymmdd hhmmss ["GMT"] [<distributions>]
   newnews newsgroups yymmddhhmmss ["GMT"] [<distributions>]
   next
   post
   slave
   stat [MessageID|Number]
   xgtitle [group_pattern]
   xhdr header [range|MessageID]
   xover [range]
   xpat header range|MessageID pat [morepat...]
   xpath MessageID
Report problems to <usenet@vlager.vbrew.com>
.
```

Tryb czytania udostępnia szereg poleceń. Wiele z nich ma ułatwić życie przeglądarkom grup dyskusyjnych. Wspomnieliśmy wcześniej, że istnieją polecenia mówiące serwerowi, by oddzielnie wysyłał nagłówek i treść artykułu. Istnieją również polecenia pokazujące listę dostępnych grup i artykułów oraz takie, które pozwalają umieszczać artykuły, czyli wysyłać je w alternatywny sposób do serwera.

Listowanie dostępnych grup

Polecenie *list* pokazuje szereg informacji różnego typu. Przede wszystkim jednak listę grup obsługiwanych przez serwer:

```
list newsgroups
215 Descriptions in form "group description".
control        News server internal group
junk           News server internal group
local.general  General local stuff
local.test     Local test group
.
```

Listowanie aktywnych grup

Polecenie *list active* pokazuje wszystkie obsługiwane grupy i podaje informacje na ich temat. Dwie liczby w każdym wierszu wyniku to górny i dolny znacznik, czyli najwyższy i najniższy numer artykułu w każdej grupie. Na ich podstawie przeglądarka może oszacować liczbę artykułów w grupie. Nieco więcej o tych numerach powiemy za chwilę. Ostatnie pole zawiera znaczniki, które kontrolują, czy wysyłanie do grupy jest dozwolone, czy grupa jest moderowana i czy wysyłane artykuły są rzeczywiście zapisywane, czy jedynie przekazywane. Znaczniki te są opisane szczegółowo w rozdziale 23. Oto przykład:

```
list active
215 Newsgroups in form "group high low flags".
control 0000000000 0000000001 y
junk 0000000003 0000000001 y
alt.test 0000000000 0000000001 y
.
```

Wysyłanie artykułu

Wspomnieliśmy, że istnieje różnica pomiędzy wysyłaniem artykułu a jego wciskaniem. Przy wciskaniu po cichu zakłada się, że artykuł już istnieje, to znaczy, że ma unikatowy identyfikator wiadomości, który został mu unikatowo przypisany przez serwer, do którego został pierwotnie wysłany i że ma pełny zestaw nagłówków. Gdy wysyłasz artykuł, tworzysz go po raz pierwszy i podajesz tylko istotne dla ciebie nagłówki, jak temat (Subject) i grupy dyskusyjne (Newsgroups), do których wysyłasz artykuł. Serwer grup dyskusyjnych, do którego wysyłasz artykuł, doda wszystkie pozostałe nagłówki i stworzy identyfikator wiadomości, który będzie używany przy umieszczaniu artykułu (wciskaniu) na innych serwerach.

Wszystko to oznacza, że wysyłanie artykułu jest prostsze, niż jego wciskanie. Przykład wysyłania może wyglądać tak:

```
post
340 Ok
From: terry@richard.geek.org.au
Subject: test message number 1
Newsgroups: junk
Body:

This is a test message, please feel free to ignore it.

.
240 Article posted
```

Wygenerowaliśmy jeszcze dwa takie artykuły, by naszym przykładom nadać cechy prawdopodobieństwa.

Listowanie nowych artykułów

Gdy przeglądarka po raz pierwszy łączy się z nowym serwerem i użytkownik wybiera grupy, które chce przeglądać, przeglądarka będzie chciała pobrać listę nowych artykułów – tych, które zostały wysłane lub odebrane od ostatniego połączenia użytkownika. Do tego celu jest używane polecenie *newnews*. Musisz podać trzy obowiązkowe argumenty: nazwę grupy lub grup, datę początkową i godzinę, od której ma być pobierana lista. Data i czas są podawane w postaci liczb sześciocyfrowych, gdzie najbardziej znacząca informacja musi być podana jako pierwsza, odpowiednio: `rrmmdd` i `ggmmss`.

```
newnews junk 990101 000000
230 New News follows
<7g2o5r$aa$6@news.vbrew.com>
<7g5bhm$8f$2@news.vbrew.com>
<7g5bk5$8f$3@news.vbrew.com>
.
```

Wybór grupy, na której mają być wykonywane operacje

Gdy użytkownik wybierze grupę do przeglądania, przeglądarka może poinformować serwer, że grupa została wybrana. Upraszcza to współdziałanie przeglądarki i serwera, ponieważ nie trzeba już wtedy wysyłać nazwy grupy przy każdym poleceniu. Polecenie *group* po prostu przyjmuje jako argument nazwę wybranej grupy. Wiele dalszych poleceń używa wybranej nazwy jako domyślnej, dopóki grupa nie zostanie podana jawnie.

```
group junk
211 3 1 3 junk
```

Polecenie *group* zwraca wiadomość zawierającą odpowiednio: liczbę aktywnych wiadomości, dolny znacznik, górny znacznik i nazwę grupy. Zapamiętaj, że choć w naszym przykładzie liczba wiadomości i górny znacznik mają tę samą wartość, to nie zawsze tak jest. W aktywnym serwerze grup artykuły wygasają lub są usuwane, co zmniejsza liczbę aktywnych wiadomości, ale górny znacznik pozostaje nietknięty.

Listowanie artykułów w grupie

Aby dostać się do artykułów w grupie, przeglądarka musi znać numery artykułów aktywnych. Polecenie *listgroup* daje listę numerów aktywnych artykułów w bieżącej lub jawnie podanej grupie:

```
listgroup junk
211 Article list follows
1
2
3
.
```

Pobieranie jedynie nagłówka artykułu

Użytkownik musi coś wiedzieć o artykule, aby mógł zdecydować, czy chce go przeczytać. Wspomnieliśmy wcześniej, że niektóre polecenia pozwalają przesyłać oddzielnie nagłówek i treść artykułu. Polecenie *head* jest używane do przesyłania do przeglądarki jedynie nagłówka zadanego artykułu. Jeżeli użytkownik nie chce czytać tego artykułu, nie marnujemy czasu ani przepustowości sieci na niepotrzebne przesyłanie jego treści, która może być duża.

Do artykułów można się odwoływać przez ich numer (uzyskany poleceniem *listgroup*) lub przez identyfikator wiadomości:

```
head 2
221 2 <7g5bhm$8f$2@news.vbrew.com> head
Path: news.vbrew.com!not-for-mail
From: terry@richard.geek.org.au
Newsgroups: junk
Subject: test message number 2
Date: 27 Apr 1999 21:51:50 GMT
Organization: The Virtual brewery
Lines: 2
Message-ID: <7g5bhm$8f$2@news.vbrew.com>
NNTP-Posting-Host: localhost
X-Server-Date: 27 Apr 1999 21:51:50 GMT
```

```
Body:
Xref: news.vbrew.com junk:2
.
```

Pobieranie jedynie treści artykułu

Jeżeli jednak użytkownik zdecyduje się, że chce przeczytać artykuł, przeglądarka potrzebuje sposobu na przesłanie samej jego treści. Do tego celu jest używane polecenie *body*. Działa w ten sam sposób co *head*, ale zwracana jest treść artykułu:

```
body 2
222 2 <7g5bhm$8f$2@news.vbrew.com> body
This is another test message, please feel free to ignore it too.

.
```

Czytanie artykułu z grupy

Choć zwykle bardziej efektywne jest oddzielne przesyłanie nagłówków i treści wybranych artykułów, czasem zdarza się, że lepiej jest przesłać pełny artykuł. Jednym z przykładów takiego zastosowania jest chęć przesłania wszystkich artykułów bez żadnej wstępnej selekcji, czyli na przykład gdy używamy programu pamięci podręcznej NNTP jak *leafnode**.

Oczywiście NNTP pozwala na takie przesyłanie i co nie jest zaskoczeniem, działa ono tak samo dobrze jak polecenie *head*. Polecenie *article* także przyjmuje numer artykułu lub ID wiadomości, ale zwraca cały artykuł włącznie z nagłówkiem:

```
article 1
220 1 <7g2o5r$aa$6@news.vbrew.com> article
Path: news.vbrew.com!not-for-mail
From: terry@richard.geek.org.au
Newsgroups: junk
Subject: test message number 1
Date: 26 Apr 1999 22:08:59 GMT
Organization: The Virtual brewery
Lines: 2
Message-ID: <7g2o5r$aa$6@news.vbrew.co>
NNTP-Posting-Host: localhost
X-Server-Date: 26 Apr 1999 22:08:59 GMT
Body:
Xref: news.vbrew.com junk:1

This is a test message, please feel free to ignore it.

.
```

Jeżeli spróbujesz pobrać nieznany artykuł, serwer zwróci ci go wraz z odpowiednim kodem odpowiedzi i być może czytelnym komunikatem tekstowym:

```
article 4
423 Bad article number
```

W tym podrozdziale omówiliśmy, jak działają najważniejsze polecenia NNTP. Jeżeli interesuje cię tworzenie oprogramowania wykorzystującego ten protokół, powinieneś

* *leafnode* jest dostępny z anonimowego serwera FTP **wpxx02.toxi.uni-wuerzburg.de** w katalogu */pub*.

skorzystać z odpowiednich dokumentów RFC. Zawierają one wiele szczegółów, których nie możemy tu taj opisać.

Przyjrzyjmy się teraz jak NNTP działa w serwerze **nntpd**.

Instalowanie serwera NNTP

Serwer NNTP (**nntpd**) może być skompilowany na dwa sposoby, w zależności od oczekiwanego obciążenia systemu grup. Nie są dostępne wersje skompilowane, ponieważ pewne wartości związane z ośrodkiem są na sztywno zaszyte w kodzie wykonywalnym. Cała konfiguracja jest realizowana przez makra zdefiniowane w pliku *common/conf.h*.

nntpd można konfigurować zarówno jako samodzielny serwer uruchamiany w czasie inicjacji systemu z pliku *rc*, jak i jako demona zarządzanego przez *inetd*. W tym drugim przypadku musisz mieć w pliku */etc/inetd.conf* następujący wpis:

```
nntp stream tcp nowait news /usr/etc/in.nntpd nntpd
```

Składnia *inetd.conf* jest szczegółowo opisana w rozdziale 12, *Ważne funkcje sieciowe*. Jeżeli konfigurujesz *nntpd* jako samodzielny serwer, pamiętaj, aby zakomentować odpowiedni wiersz w pliku *inetd.conf*. W obu przypadkach pamiętaj, by w */etc/services* pojawił się następujący wiersz:

```
nntp   119/tcp readnews untp # Network News Transfer Protocol
```

Aby tymczasowo zapisać jakieś artykuły przychodzące, *nntpd* potrzebuje katalogu *.tmp* w twoim katalogu buforowym grup dyskusyjnych. Powinieneś go stworzyć, używając poniższych poleceń:

```
# mkdir /var/spool/news/.tmp
# chown news.news /var/spool/news/.tmp
```

Ograniczanie dostępu NNTP

Dostęp do zasobów NNTP jest zarządzany przez plik *nntp_access* znajdujący się w katalogu */etc/news*. Wiersze tego pliku opisują prawa dostępu udzielane obcym hostom. Każdy wiersz ma następujący format:

```
o□rodek   read|xfer|both|no   post|no   [!bezgrup]
```

Jeżeli klient łączy się z portem NNTP, **nntpd** próbuje uzyskać jego pełną nazwę domenową na podstawie adresu IP. Nazwa hosta klienta i jego adres IP są sprawdzane z polem `o□rodek` każdego wpisu w kolejności, w jakiej pojawiają się w pliku. Dopasowania nie może być pełne lub częściowe. Jeżeli wpis pasuje dokładnie, jest realizowany. Jeżeli dopasowanie jest częściowe, zadziała tylko wtedy, gdy nie ma innych, lepszych (lub przynajmniej równie dobrych) dopasowań. `o□rodek` może być podany w jednej z następujących postaci:

Nazwa hosta
: Jest to pełna nazwa domenowa hosta. Jeżeli jest w pełni zgodna z nazwą kanoniczną hosta klienta, wpis jest stosowany, a wszystkie następne są zignorowane.

Adres IP
: Jest to adres IP zapisany w postaci liczbowej. Jeżeli adres klienta jest z nim zgodny, wpis jest stosowany, a wszystkie następne są zignorowane.

Nazwa domeny
: Jest to nazwa domeny podana w postaci `*.domena`. Jeżeli jest zgodna z nazwą domeny klienta, wpis jest stosowany.

Nazwa sieci
: Jest to nazwa sieci zgodna z opisem w pliku */etc/networks*. Jeżeli numer IP klienta pasuje do numeru sieci związanego z nazwą sieci, wpis jest stosowany.

Wartość domyślna
: Do ciągu `default` pasuje dowolny klient.

Wpisy z bardziej ogólną specyfikacją ośrodka powinny być podane wcześniej, ponieważ wszelkie dopasowania zostaną zastąpione dokładniejszymi dopasowaniami występującymi dalej.

Drugie i trzecie pole opisują prawa dostępu udzielone klientowi. Drugie pole opisuje szczegółowe prawa niezbędne do pobrania artykułu przez ściągnięcie (`read`) i jego wrzucenie przez wciśnięcie (`xfer`). Wartość `both` zawiera oba poprzednie, a `no` oznacza całkowity zakaz dostępu. Trzecie pole daje klientowi prawo do wysyłania artykułów, czyli do ich umieszczania bez pełnej informacji w nagłówku, która jest uzupełniana przez oprogramowanie do obsługi grup. Jeżeli drugie pole zawiera `no`, trzecie pole jest ignorowane.

Czwarte pole jest opcjonalne i zawiera oddzielaną przecinkami listę grup, do których klient nie ma dostępu.

Oto przykładowy plik *nntp_access*:

```
#
# domy□lnie ka□dy mo□e przesy□a□ artyku□y, ale nie ka□dy mo□e
# je czyta□ lub pisa□ nowe
default            xfer      no
#
# public.vbrew.com oferuje dost□p przez modem. Pozwalamy na
# czytanie i wysy□anie artyku□ów do wszystkich grup poza
# local.*
public.vbrew.com   read      post      !local
#
# wszystkie pozosta□e hosty w browarze mog□ czyta□ i wysy□a□
*.vbrew.com        read      post
```

Autoryzacja NNTP

Demon *nntpd* oferuje prosty schemat autoryzacji. Jeżeli jakieś leksemy opisujące dostęp w pliku *nntp_access* napiszesz dużymi literami, *nntpd* zażąda autoryzacji klienta dla danej operacji. Na przykład, gdybyś zapisał prawa dostępu jako `Xfer` `XFER` (zamiast `xfer`), *nntpd* nie pozwoliłby klientowi przesłać artykułów bez autoryzacji.

Procedura autoryzacji jest zaimplementowana przez nowe polecenie NNTP: *AUTHINFO*. W tym poleceniu klient przesyła nazwę użytkownika i hasło do serwera

NNTP. Demon *nntpd* sprawdza je z plikiem */etc/passwd*, aby dowiedzieć się, czy użytkownik należy do grupy **nntp**.

Aktualna implementacja autoryzacji NNTP ma charakter eksperymentalny i dlatego nie została zaimplementowana jako przenośna. Działa więc tylko z bazą czystych haseł – hasła shadow nie są rozpoznawane. Jeżeli kompilujesz źródła i masz zainstalowany pakiet PAM, bardzo łatwo zmienić procedurę sprawdzania haseł.

Współpraca nntpd z C News

Gdy *nntpd* odbierze artykuł, musi go dostarczyć do podsystemu grup. W zależności od tego, czy został on odebrany poleceniem *IHAVE* czy *POST*, jest przekazywany odpowiednio do *rnews* lub *inews*. Zamiast wywoływać *rnews*, możesz także skonfigurować (w czasie kompilacji) wywoływanie przetwarzania wsadowego przychodzących artykułów i przenosić uzyskane wsady do katalogu */var/spool/news/in.coming*, gdzie oczekują na pobranie przez *relaynews* przy następnym przebiegu kolejki.

nntpd musi mieć dostęp do pliku *history*, by móc poprawnie obsługiwać protokół *ihave/sendme*. W czasie kompilacji musisz podać dokładną ścieżkę do tego pliku. Jeżeli używasz C News, sprawdź, czy C News i *nntpd* są zgodne co do formatu pliku historii. C News przy dostępie używa funkcji mieszającej *dbm*. Jednak istnieje szereg różnych, niezbyt kompatybilnych implementacji biblioteki *dbm*. Jeżeli C News został skonsolidowany z jakąś inną wersją biblioteki *dbm*, która nie jest zgodna z wersją znajdująca się w twojej standardowej bibliotece *libc*, musisz skonsolidować *nntpd* z tą samą biblioteką.

Niezgodności pomiędzy *nntpd* i C News są czasem powodem generowania komunikatów o błędach w logu systemowym, mówiących o tym, że *nntpd* nie może go poprawnie otworzyć. Może się też zdarzyć, że zobaczysz podwójne artykuły odebrane przez NNTP. Dobrym testem na błędne funkcjonowanie przesyłania grup jest pobranie artykułu z obszaru buforowego, wykonanie telnet na port **nntp** i zaoferowanie go *nntpd* zgodnie z tym, co pokazano w przykładzie poniżej. Oczywiście musisz zastąpić `msg@id` ID wiadomości, którą chcesz przekazać do *nntpd*:

```
$ telnet localhost nntp
Trying 127.0.0.1...
Connected to localhost
Escape characeters is '^]'.
201 vstout NNTP[auth] server version 1.5.11t (16 November 1991) ready at Sun
Feb 6 16:02:32 1194 (no posting)
IHAVE msg@id
435 Got it.
QUIT
```

Ta konwersacja pokazuje poprawną reakcję *nntpd*. Komunikat `Got It` mówi, że artykuł już istnieje. Gdybyś zamiast niego dostał komunikat *335 Ok*, oznaczałoby to, że przeszukiwanie pliku historii z jakiegoś powodu się nie powiodło. Zakończ konwersację wpisując [Ctrl+D]. W logu systemowym możesz sprawdzić, co poszło źle. *nntpd* zapisuje do logu wszelkie komunikaty, używając funkcji *syslog:* `daemon`. `Niekompatybilna biblioteka` *dbm* zwykle sama zgłasza komunikat mówiący, że wywołanie *dbminit* się nie powiodło.

23

Internet News



Demon Internet News (INN) jest prawdopodobnie najpopularniejszym z obecnie używanych serwerów grup dyskusyjnych. Jest bardzo elastyczny i odpowiedni dla wszystkich ośrodków udostępniających grupy, może poza najmniejszymi*. INN doskonale się skaluje i jest przystosowany do dużych ośrodków grup dyskusyjnych.

Serwer INN składa się z szeregu elementów, z których każdy ma własne pliki konfiguracyjne. Omówimy je wszystkie kolejno. Konfiguracja INN-a może być nieco absorbująca, ale w tym rozdziale opiszemy wszystkie etapy i podamy wystarczająco dużo informacji, byś mógł zrozumieć strony podręcznika INN i jego dokumentację oraz stworzyć konfiguracje dla dowolnych zastosowań.

Pewne tajniki wewnętrzne INN-a

Rdzeniem INN-a jest demon *innd*. Jego zadaniem jest obsługa wszystkich przychodzących artykułów, zachowywanie ich lokalnie i dalsze przekazywanie, o ile jest taka potrzeba. Jest uruchamiany w czasie inicjacji systemu i działa jako proces w tle. Działanie w trybie demona jest wydajniejsze, ponieważ pliki stanu są czytane tylko raz, przy uruchomieniu. W zależności od wielkości obsługiwanych przez ciebie grup, pewne pliki, takie jak *history* (zawierający listę ostatnio przetworzonych artykułów), mogą zajmować od kilku do kilkudziesięciu megabajtów.

Inną ważną funkcją INN-a jest to, że zawsze działa tylko jedno jego wcielenie. Ma to także duży wpływ na wydajność, ponieważ demon może przetwarzać wszystkie artykuły bez martwienia się o synchronizację stanów wewnętrznych z innymi

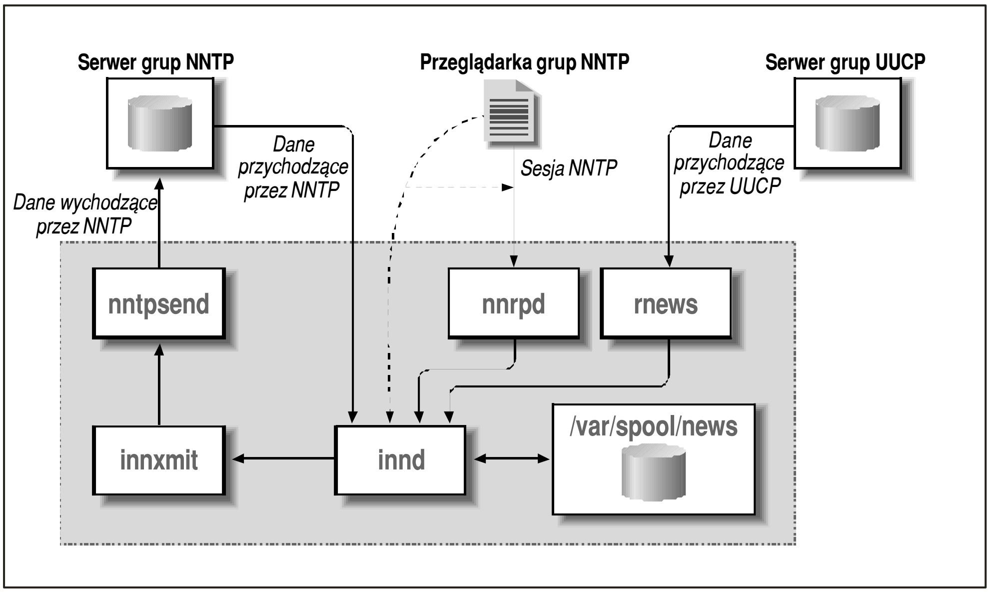
* Dla bardzo małych ośrodków lepiej nadaje się program pamięci podręcznej NNTP, jak *leafnode*, dostępny pod adresem *http://wpxx02.toxi.uni-wuerzburg.de/~krasel/leadnode.html*.

kopiami *innd* dostającymi się do bufora grup w tym samym czasie. Jednak taka konstrukcja ma wpływ na całościową architekturę systemu grup, ponieważ chodzi o to, aby przychodzące wiadomości były przetwarzane tak szybko, jak to możliwe, i jest nie do przyjęcia, by serwer zajmował się tak przyziemnymi zadaniami, jak obsługa wiadomości przychodzących przez UUCP. Dlatego te zadania zostały oddzielone od głównego serwera i zaimplementowane w oddzielnych programach pomocniczych. Rysunek 23-1 próbuje pokazać powiązania pomiędzy *innd* a innymi lokalnymi zadaniami, zdalnymi serwerami i przeglądarkami grup dyskusyjnych.

Obecnie do przesyłania artykułów najczęściej służy NNTP, a *innd* bezpośrednio obsługuje tylko ten protokół. Oznacza to, że *innd* oczekuje na gnieździe TCP (port 119) na połączenia i przyjmuje artykuły, używając protokołu ihave.

Artykuły przybywające inną drogą, niż przez NNTP, są obsługiwane pośrednio przez inny proces przyjmujący artykuły i przekazujący je do *innd* przez NNTP. Wsady przychodzące na przykład przez łącze UUCP są tradycyjnie obsługiwane przez program *rnews*. Wersja tego programu zawarta w pakiecie INN w razie potrzeby dekompresuje wsady i dzieli je na pojedyncze artykuły. Następnie po kolei przesyła je do *innd*.

Przeglądarki grup mogą dostarczać wiadomości, gdy użytkownik wyśle artykuł. Ponieważ obsługa przeglądarek zasługuje na specjalną uwagę, wrócimy do niej za chwilę.



Rysunek 23-1. Uproszczony schemat architektury INN-a

Przyjmując artykuł, *innd* najpierw sprawdza jego ID w pliku *history*. Zduplikowane artykuły są odrzucane, a ich pojawienie się jest (opcjonalnie) odnotowywane. To sa-

mo do tyczy artykułów, które są zbyt stare lub brakuje im wymaganych pól nagłówka, takich jak `Subject:`*. Jeżeli *innd* stwierdzi, że artykuł jest do przyjęcia, sprawdza wiersz nagłówka `Newsgroups:`, by stwierdzić, do której grupy został wysłany artykuł. Jeżeli w pliku *active* znajdzie jedną lub więcej grup, artykuł jest zapisywany w postaci pliku na dysku. W przeciwnym razie jest przesyłany do specjalnej grupy *junk*.

Pojedyncze artykuły są przechowywane w katalogu */var/spool/news*, zwanym także *buforem grup*. Każda grupa ma oddzielny katalog, w którym artykuł jest zapisywany jako oddzielny plik. Nazwy plików mają postać kolejnych numerów, a więc na przykład artykuł z grupy *comp.risks* może być zapisany jako *comp/risks/217*. Gdy *innd* stwierdzi, że nie istnieje katalog, w którym trzeba zapisać artykuł, automatycznie go tworzy.

Zapewne zechcesz też przekazywać artykuły dalej, jako dane wychodzące, a nie tylko zapisywać je lokalnie. Zarządza tym plik *newsfeeds*, który opisuje wszelkie ośrodki, do których powinny być wysyłane artykuły z danej grupy.

Podobnie jak po stronie odbiorczej *innd*, tak i po stronie wychodzącej, przetwarzanie jest obsługiwane także przez jeden interfejs. Zamiast samodzielnie obsługiwać wszelkie specyficzne sposoby transportu, *innd* opiera się na różnych ukrytych systemach zarządzających przesyłaniem artykułów do innych serwerów grup. Grupy wychodzące są obsługiwane przez *kanały*. W zależności od przeznaczenia kanał może mieć różne atrybuty, określające dokładnie, jakie informacje przekazuje do niego *innd*.

W przypadku danych wychodzących przez NNTP, *innd* mógłby przy uruchamianiu wywołać program *innxmit* i przekazywać mu na standardowe wejście ID, rozmiar i nazwę pliku każdego artykułu, który powinien być wysłany dalej. Natomiast w przypadku danych wychodzących przez UUCP, mógłby zapisywać rozmiar artykułu i jego nazwę pliku do specjalnego pliku log, który byłby sprawdzany w regularnych odstępach czasu przez inny proces, który tworzyłby wsady i kolejkował je w podsystemie UUCP.

Poza tymi dwoma przykładami, istnieją inne typy kanałów, które niekoniecznie dotyczą danych wychodzących. Są one używane na przykład przy archiwizowaniu pewnych grup lub przy generowaniu informacji przeglądowych. Informacje takie mają pomagać przeglądarkom efektywniej dzielić artykuły na wątki. Przeglądarki starego typu muszą przeglądać kolejno wszystkie artykuły, by uzyskać z nagłówka informacje wymagane do podziału na wątki. Obciąża to poważnie serwer, szczególnie jeżeli używasz NNTP. Co więcej jest to bardzo wolne**. Mechanizm informacji poglądowych łagodzi ten problem, ponieważ zapisuje wstępnie wszystkie istotne nagłówki każdej grupy w oddzielnym pliku (*.overview*). Później przeglądarka może pobrać te informację albo bezpośrednio ją odczytując z katalogu bufora, albo wykonując polecenie *XOVER* przy połączeniu przez NNTP. Demon *innd* przekazuje

---

* Wiek pokazuje pole nagłówka `Date:`. Ograniczenie zwykle wynosi dwa tygodnie.

** Podział tysiąca artykułów na wątki przy komunikacji z obciążonym serwerem może potrwać 15 minut, co jest do przyjęcia tylko dla nałogowców uzależnionych od Usenetu.

wszystkie artykuły poleceniu *overchan*, które jest połączone z demonem przez kanał. Dalej, przy okazji omawiania konfiguracji dostarczania grup, zobaczymy, jak to jest realizowane.

Przeglądarki grup dyskusyjnych i INN

Przeglądarki grup, działające na tej samej maszynie co serwer (lub mające zamontowany bufor grup serwera przez NFS) mogą czytać artykuły bezpośrednio z katalogów bufora. W celu wysłania artykułu stworzonego przez użytkownika, wywołują program *inews*, który dodaje brakujące pola nagłówka i przekazuje go do demona przez NNTP.

Ewentualnie przeglądarki mogą dostawać się do serwera zdalnie przez NNTP. Aby uniknąć obciążenia demona, ten typ połączenia jest obsługiwany inaczej niż dostarczanie grup oparte na NNTP. Gdy przeglądarka podłączy się do serwera NNTP, *innd* tworzy oddzielny program *nnrpd* obsługujący sesję, natomiast *innd* wraca do robienia ważniejszych rzeczy (na przykład odbierania przychodzących wiadomości)*. Zastanawiasz się pewnie, jak proces *innd* rozróżnia przychodzące wiadomości od podłączającej się przeglądarki grup. Odpowiedź jest prosta: protokół NNTP wymaga, by przeglądarka oparta na NNTP wysłała polecenie *mode reader* po połączeniu się z serwerem. Gdy polecenie to zostanie odebrane, serwer uruchamia *nnrpd*, przekazuje mu połączenie i powraca do nasłuchiwania połączeń z innych serwerów grup. Znana jest przynajmniej jedna przeglądarka DOS-owa, która nie jest skonfigurowana w ten sposób i nie uda je się jej połączyć z INN, ponieważ sam *innd* nie rozpozna jeżadnych poleceń używanych do czytania grup, jeśli nie wie, że połączenie pochodzi od przeglądarki.

Nieco więcej o dostępie przeglądarki do INN-a powiemy w dalszej części tego rozdziału: *Kontrolowanie dostępu przeglądarki*.

Instalowanie INN-a

Zanim zagłębimy się w konfigurację INN-a, powiemy trochę o jego instalacji. Przeczytaj ten podrozdział, nawet jeżeli zainstalowałeś już INN-a z jakąś dystrybucją Linuksa. Znajdziesz tu pewne wskazówki dotyczące bezpieczeństwa i kompatybilności.

Dystrybucje Linuksa od pewnego czasu zawierają Version INN-1.4sec. Niestety ta wersja wnosi dwa problemy związane z bezpieczeństwem. Nowsze wersje nie stwarzają już tych problemów, a większość dystrybucji Linuksa zawiera skompilowane pliki binarne wersji 2. INN-a (lub nowszych).

Jeżeli chcesz, możesz samodzielnie skompilować INN-a. Kod źródłowy można zdobyć z **ftp.isc.org** z katalogu */isc/inn/*. Kompilacja INN-a wymaga edycji pliku konfiguracyjnego, który przekazuje INN-owi pewne szczegóły na temat systemu operacyjnego i pewnych funkcji, które mogą wymagać niewielkich modyfikacji.

* Nazwa programu *nnrpd* pochodzi od słów „Net News Read & Post Daemon".

Kompilacja samego pakietu jest prosta. Zawiera on bowiem skrypt *BUILD*, który przeprowadzi cię przez cały proces. Kod źródłowy zawiera także szczegółową dokumentację, mówiącą, jak zainstalować i skonfigurować INN-a.

Po zainstalowaniu wszystkich plików binarnych, mogą być potrzebne pewne ręczne poprawki zapewniające kompatybilność INN-a z różnymi innymi aplikacjami, które mogą wymagać dostępu do programów *rnews* lub *inews*. Na przykład UUCP spodziewa się programu *rnews* w katalogu */usr/bin* lub */bin*, natomiast INN instaluje go domyślnie w */usr/lib/bin*. Sprawdź, czy */usr/lib/bin/* jest w domyślnej ścieżce przeszukiwań lub czy istnieje dowiązanie symboliczne wskazujące na rzeczywistą lokalizację poleceń *rnews* i *inews*.

Podstawowe konfigurowanie INN-a

Jedną z największych trudności, na jaką może natrafić początkujący, jest to, że INN do poprawnego funkcjonowania wymaga działającej konfiguracji sieciowej, nawet gdy operuje na samodzielnym hoście. Dlatego trzeba dopilnować dwóch spraw. Po pierwsze, jądro twojego Linuksa musi obsługiwać sieć TCP/IP, gdy chcesz uruchamiać INN-a. Po drugie, musisz mieć skonfigurowany interfejs pętli zwrotnej, opisany w rozdziale 5, *Konfigurowanie sieci TCP/IP*.

Następnie trzeba sprawdzić, czy *innd* jest uruchamiany w czasie inicjacji komputera. Domyślna instalacja INN-a zawiera skrypt o nazwie *boot* w katalogu */etc/news/*. Jeżeli twoja dystrybucja używa pakietu *init* typu System V, wystarczy, że stworzysz dowiązanie symboliczne do pliku */etc/init.d/inn* tak, by wskazywało na */etc/news/boot*. W innych wersjach *init* musisz sprawdzić, czy */etc/news/boot* jest uruchamiany z jednego z twoich skryptów *rc*. Ponieważ INN wymaga sieci, skrypt startowy powinien być uruchamiany *po* skonfigurowaniu interfejsów sieciowych.

Pliki konfiguracyjne INN-a

Jeżeli wykonałeś te podstawowe zadania, możesz teraz przejść do naprawdę ciekawej części INN-a: jego plików konfiguracyjnych. Wszystkie te pliki znajdują się w katalogu */etc/news*. W plikach konfiguracyjnych wersji 2. zostały wprowadzone pewne zmiany, a tu opisujemy właśnie tę wersję. Jeżeli pracujesz ze starszą wersją, ten rozdział powinien ci pomóc w uaktualnieniu konfiguracji. W kilku kolejnych podrozdziałach omówimy kolejno pliki, tworząc przykładową konfigurację dla browaru wirtualnego.

Gdybyś chciał dowiedzieć się więcej na temat funkcji poszczególnych plików konfiguracyjnych, możesz także poczytać poświęcone im strony podręcznika elektronicznego, zawarte w dystrybucji INN-a.

Parametry globalne

Istnieje szereg parametrów, które mają znaczenie globalne. Dotyczą one wszystkich grup.

Plik inn.conf

Głównym plikiem konfiguracyjnym INN-a jest *inn.conf*. Między innymi określa on nazwę, pod jaką twoja maszyna jest znana w sieci Usenet. Wersja 2. INN-a pozwala skonfigurować w tym pliku zdumiewającą wiele parametrów. Na szczęście większość ma wartości domyślne, które są sensowne dla prawie wszystkich ośrodków. Plik *inn.conf(5)* opisuje szczegółowo wszystkie parametry i powinieneś go dokładnie przeczytać, jeżeli napotkasz jakiekolwiek problemy.

Prosty przykładowy plik *inn.conf* mógłby wyglądać tak:

```
# Przyk□adowy inn.conf dla browaru wirtualnego
server:            vlager.vbrew.com
domain:            vbrew.com
fromhost:          vbrew.com
pathhost:          news.vbrew.com
organization:      The Virtual Brewery
mta:               /usr/sbin/sendmail -oi %s
moderatormailer:   %s@uunet.uu.net
#
# □cie□ki do komponentów i plików INN-a
#
pathnews:          /usr/lib/news
pathbin:           /usr/lib/news/bin
pathfilter:        /usr/lib/news/bin/filter
pathcontrol:       /usr/lib/news/bin/control
pathdb:            /var/lib/news
pathetc:           /etc/news
pathrun:           /var/run/news
pathlog:           /var/log/news
pathhttp:          /var/log/news
pathtmp:           /var/tmp
pathspool:         /var/spool/news
patharticles:      /var/spool/news/articles
pathoverview:      /var/spool/news/overview
pathoutgoing:      /var/spool/news/outgoing
pathincoming:      /var/spool/news/incoming
patharchive:       /var/spool/news/archive
pathuniover:       /var/spool/news/uniover
overviewname:      .overview
```

Pierwszy wiersz mówi programom *rnews* i *inews*, z którymi hostami mają się kontaktować, aby dostarczać artykuły. Ten wpis jest bezwzględnie konieczny. Aby przekazać artykuły do *innd*, musi zostać nawiązane połączenie NNTP z serwerem.

Słowo kluczowe `domain` powinno określać domenę pełnej nazwy domenowej hosta. Kilka programów potrzebuje tej domeny. Jeżeli twoja biblioteka resolvera zwraca jedynie nazwę hosta, jest do niego dokleјana właśnie ta domena. Lepiej więc zdefiniować `domain`, tym bardziej, że nie jest to trudne.

Następny wiersz definiuje nazwę hosta, z której korzysta *inews*, kiedy dodaje pola `From:` do artykułów wysłanych przez użytkowników lokalnych. Większość przeglądarek grup używa pola `From:` do tworzenia odpowiedzi do autora artykułu. Jeżeli pominiesz to pole, jego domyślna wartość zostanie ustalona na podstawie pełnej nazwy domenowej twojego hosta. Nie zawsze jest to najlepsze rozwiązanie. Może się zdarzyć na przykład, że wiadomości i poczta są obsługiwane przez

różne hosty. W takiej sytuacji możesz podać pełną nazwę domenową hosta pocztowego podyrektywie `fromhost`.

Wiersz `pathhost` definiuje nazwę hosta INN, która jest dodawana do pola `Path:` przy odbieraniu artykułu. Zwykle będziesz chciał używać pełnej nazwy domenowej twojego serwera grup. W takiej sytuacji możesz pominąć to pole, ponieważ takie jest ustawienie domyślne. Jeżeli obsługujesz dużą domenę, ze chcesz czasem użyć nazwy ogólnej, jak **news.vbrew.com**. Ułatwi ci to przeniesienie systemu grup dyskusyjnych na innego hosta, jeżeli kiedyś zajdzie taka potrzeba.

Następny wiersz zawiera słowo kluczowe `organization`. Ta dyrektywa pozwala na konfigurowanie napisu, jaki *inews* umieści w wierszu `Organization:` w artykułach wysyłanych przez użytkowników lokalnych. Powinieneś tam umieścić opis twojej firmy lub jej pełną nazwę. Możesz jednak nie być tak oficjalny i przedstawić się bardziej dowcipnie, co jest teraz modne.

Wpis `moderatormailer` definiuje domyślny adres używany, gdy użytkownik próbuje wysłać artykuł do grupy moderowanej. Lista adresów moderatorów dla każdej grupy zwykle znajduje się w oddzielnym pliku, ale jej aktualizowanie wymaga niemało pracy (i czasu). Dla tego go wpis `moderatormail` jest używany w ostateczności. Jeżeli jest zdefiniowany, *inews* zastąpi ciąg `%s` (nieco go zmieniając) nazwą grupy i wyśle cały artykuł na ten adres. Na przykład przy wysyłaniu do grupy *soc.feminism*, artykuł jest, zgodnie z powyższą konfiguracją, wysyłany pod adres *soc-feminism@uunet.uu.net*. W UUNET powinien być zainstalowany alias pocztowy dla każdego adresu, przekazujący automatycznie wszystkie wiadomości do odpowiedniego moderatora.

Każdy z pozostałych wpisów określa lokalizację niektórych plików związanych z komponentami lub plików wykonywalnych należących do INN. Jeżeli zainstalowałeś INN-a z pakietu, ścieżki te powinny być już skonfigurowane. Jeżeli instalujesz go ze źródeł, będziesz musiał skonfigurować je zgodnie z tym, jak zainstalowałeś INN-a.

Konfigurowanie grup dyskusyjnych

Administrator grup dyskusyjnych może kontrolować dostęp użytkowników do grup. INN zawiera dwa pliki konfiguracyjne pozwalające administratorowi pokazać, które grupy mają być obsługiwane i dodać dla nich opis.

Pliki active i newsgroups

Pliki *active* i *newsgroups* są używane do przechowywania i opisywania grup dyskusyjnych obsługiwanych przez dany serwer. Zawierają spis grup, które chcemy otrzymywać i do których chcemy wysyłać artykuły, oraz dotyczących ich informacji administracyjnych. Pliki te znajdują się w katalogu */var/lib/news/*.

Plik *active* określa, które grupy obsługuje serwer. Jego składnia jest prosta. Każdy wiersz pliku *active* składa się z czterech pól oddzielonych białymi znakami:

```
nazwa zngór zndol znaczniki
```

Pole `nazwa` to nazwa grupy. Pole `zngór` zawiera najwyższy numer artykułu w grupie. Pole `zndol` zawiera najniższy numer aktywnego artykułu w grupie. Aby pokazać, jak to działa, rozważ następujący scenariusz. Wyobraź sobie, że mamy nowo utworzoną grupę dyskusyjną: i `zngór`, i `zndol` mają wartość 0, ponieważ w grupie nie ma artykułów. Jeśli wyślemy 5 artykułów, zostaną one ponumerowane od 1 do 5. `zngór` będzie teraz miał wartość 5, czyli najwyższy numer artykułu, a `zndol` będzie równy 1 – numerowi pierwszego artykułu. Jeżeli artykuł 5. zostanie anulowany, nie nastąpi zmiana. `zngór` będzie dalej miał wartość 5, gdyż numery artykułów nie mogą być relokowane, a `zndol` będzie miał dalej wartość 1. Jeżeli teraz anulujemy artykuł 1, `zngór` pozostanie bez zmian, a `zndol` będzie miał wartość 2, ponieważ 1 nie jest już artykułem aktywnym. Jeżeli teraz wyślemy nowy artykuł, zostanie mu przypisany numer 6, a więc `zngór` będzie teraz miał wartość 6. Artykuł 5 był wykorzystywany, a więc nie zmieniamy jego numeru. Wartość `zndol` pozostaje na poziomie 2. Mechanizm ten pozwala nam prosto alokować unikalne numery dla nowych artykułów i szacować liczbę aktywnych artykułów w grupie: `zngór-zndol`.

Ostatnie pole może zawierać jedną z następujących wartości:

y
: Dopuszczalne jest wysyłanie bezpośrednio do serwera.

n
: Wysyłanie bezpośrednio do serwera nie jest dopuszczalne. Zapobiega to wysyłaniu wiadomości przez przeglądarki bezpośrednio do serwera grup. Nowe artykuły mogą być odbierane tylko z innych serwerów grup.

m
: Grupa jest moderowana. Wszelkie artykuły wysłane do tej grupy są przekazywane do jej moderatora w celu zatwierdzenia, za nim pojawią się w grupie. Większość grup nie jest moderowana.

j
: Artykuły z tej grupy nie są przechowywane, ale jedynie przekazywane dalej. Powoduje to, że serwer grup przyjmuje artykuł, ale wszystko co z nim robi, to przekazanie dalszym serwerom grup. Artykuły nie są dostępne dla przeglądarek podłączających się do tego serwera w celu czytania grup.

x
: Artykuły nie mogą być wysyłane do tej grupy. Jedynym sposobem na dostarczenie artykułów do tego serwera jest ich przesłanie z innego serwera grup. Przeglądarki nie mogą bezpośrednio zapisywać artykułów na serwerze.

`=foo.bar`
: Artykuły są zapisywane lokalnie w grupie „foo.bar".

W naszej prostej konfiguracji serwera obsługujemy niewiele grup, a więc plik */var/lib/news/active* wygląda tak:

```
control 0000000000 0000000001 y
junk 0000000000 0000000001 y
rec.crafts.brewing 0000000000 0000000001 y
rec.crafts.brewing.ales 0000000000 0000000001 y
rec.crafts.brewing.badtaste 0000000000 0000000001 y
```

```
rec.crafts.brewing.brandy 0000000000 0000000001 y
rec.crafts.brewing.champagne 0000000000 0000000001 y
rec.crafts.brewing.private 0000000000 0000000001 y
```

Nu mery `zngór` i `zndol` w tym przykładzie mają war toś ci pier wotne, ta kie jak przy two rzen iu no wych grup. Będą one wyglądały nie co in ac zej, gdy gru pa bę dzie aktywna przez pe wien czas.

Plik *newsgroups* jest jeszcze prostszy. Zawiera jednowierszowy opis każdej grupy. Nie któ re przeglą dar ki mogą od czy ty wać i przed sta wiać za war te w nim in for macje użyt kow ni ko wi, po ma gając mu w ten spo sób zde cy do wać, do któ rej gru py się zapisać.

For mat pli ku *newsgroups* jest pro sty:

```
nazwa opis
```

Po le `nazwa` to na zwa gru py, a `<opis` to wiersz z in formacją o treś ci gru py.

Chce my za pis ać się do po niżs zych grup, obsługiw any ch przez nasz ser wer, a więc two rzymy na stęp ujący plik *newsgroups*:

```
rec.crafts.brewing.ales       Home brewing Ales and Lagers
rec.crafts.brewing.badtaste   Home brewing foul tasting brews
rec.crafts.brewing.brandy     Home brewing your own Brandy
rec.crafts.brewing.champagne  Home brew your own Champagne
rec.crafts.brewing.private    The Virtual Brewery home brewers group
```

Konfigurowanie dostarczania grup do innych serwerów

INN zapewnia administratorowi grup możliwość kontroli nad tym, które grupy i w ja ki spo sób są prze ka zy wa ne do in nych ser we rów. Naj po pu lar niej sze me to dy wy ko rzy stują opi sa ny wcze śniej pro to kół NNTP, ale INN do pusz cza ta kże in ne protokoły, na przykład UUCP.

Plik newsfeeds

Plik *newsfeeds* okre śla, gdzie będą kie ro wa ne no we ar ty kuły. Zwy kle znaj du je się on w ka ta lo gu */etc/news/*.

For mat pli ku *newsfeeds* początkowo wy daje się nie co skom plik owa ny. Opis zemy tu je go og ólny wygląd, a szczegóły znaj dziesz na stro nie podręcz nika elekt roni cznego *newsfeeds(5)*.

For mat jest na stę pujący:

```
# format pliku newsfeeds
o□rodek:wzorzec:znaczniki:parametry
o□rodek2:wzorzec2\
:znaczniki2:parametry2
```

Ka ż de połącze nie zwią za ne z prze syłaniem grup do ośrod ka jest opi sa ne w jed nym wierszu lub w kilku (wtedy na końcu kontynuowanego wiersza trzeba umieścić znak kon ty nu acji \). Znak: roz dzie la po la. Znak # na początku wier sza ozna cza komentarz.

Po le `o□rodek` za wie ra na zwę ośrod ka, dla któ re go jest prze zna czo na da na por cja grup. Na zwy mogą być za pi sy wa ne w do wol ny spo sób i nie mu si być to na zwa do-

menowa. Nazwa ta zostanie użyta później do wskazania wpisu w tablicy z nazwą hosta, która jest potrzebna programowi *innxmit* wysyłającemu artykuły przez NNTP do serwera zdalnego. Możesz mieć po kilka wpisów dla każdego ośrodka. Każdy wpis będzie rozpatrywany indywidualnie.

Pole `wzorzec` określa, które grupy mają być wysyłane do danego ośrodka. Domyślnie wysyłane są wszystkie grupy, a więc jeżeli tego właśnie chcesz, po prostu pozostaw pole puste. Zwykle pole to zawiera oddzielaną przecinkami listę wyrażeń-wzorców. Do znaku * pasuje zero lub więcej dowolnych znaków, znak (.) nie ma szczególnego znaczenia, znak ! (jeżeli został użyty na początku wyrażenia) realizuje logiczną operację NOT, a znak @ na początku nazwy grupy oznacza „Nie przekazuj żadnych artykułów, które zostały wysłane do tej grupy”. Lista jest odczytywana i analizowana od lewej do prawej strony, a więc powinieneś na początku umieszczać dokładniejsze reguły. Wzorzec:

```
rec.crafts.brewing*,!rec.crafts.brewing.poison,@rec.crafts.brewing.private
```

spowoduje wysłanie wszystkich grup z hierarchii *rec.crafts.brewing* z *wyjątkiem* grupy *rec.crafts.brewing.poison*. Nie zostaną przekazane żadne artykuły wysłane do grupy *rec.crafts.brewing.private*. Zostaną zatrzymane i będą dostępne tylko dla ludzi z tego serwera. Gdybyś zamienił miejscami dwa pierwsze wzorce, pierwszy zostałby nadpisany przez drugi i skończyłoby się to przekazaniem wszystkich artykułów z grupy *rec.crafts.brewing.poison*. To samo dotyczy pierwszego i ostatniego wzorca. Zawsze musisz umieszczać dokładniejsze wzorce przed mniej dokładnymi, jeżeli mają działać tak, jak chcesz.

Pole `znaczniki` kontroluje przekazywanie artykułów do danego ośrodka i nakłada ograniczenia. Pole to jest oddzielaną przecinkami listą zawierającą niżej wymienione elementy:

`<rozmiar`
: Artykuł musi mieć mniej bajtów niż zadany rozmiar.

`Aelementy`
: Sprawdzanie artykułów. `Elementy` mogą być mieć wartość jednego lub kilku `d` (musi mieć nagłówek `Distribution`) lub `p` (nie sprawdzaj nagłówka `Path` dla tego ośrodka).

`Bwys/nis`
: Rozmiar bufora wewnętrznego przed zapisaniem na wyjście.

`H/liczba/`
: Artykuł musi mieć mniej niż `liczba` hopów – domyślnie 1.

`Irozmiar`
: Rozmiar bufora wewnętrznego (do przesyłania plików).

`Mwzorzec`
: Do tego wzorca pasują tylko grupy moderowane.

`Nwzorzec`
: Do tego wzorca pasują tylko grupy niemoderowane.

`S`*rozmiar*

Rozpoczęcie buforowania, jeżeli zostanie zakolejkowane więcej bajtów niż zadany rozmiar.

`T`*typ*

Typy przekazywania: `f` (plik), `m` (strumień; pole *parametry* musi zawierać nazwy wpisów, do których artykuły będą przekazywane), `p` (przekazywanie przez potok do programu), `c` (wysyłanie do kanału stdin podprocesu pola *parametry*) i `x` (jak `c`, ale obsługuje polecenia na stdin).

`W`*elementy*

Co zapisać: `b` (rozmiar artykułu w bajtach), `f` (pełna ścieżka), `g` (pierwsza grupa dyskusyjna), `m` (ID wiadomości), `n` (ścieżka względna), `s` (ośrodek, z którego przyszedł artykuł), `t` (czas odebrania), `*` (nazwy strumieni wejściowych lub wszystkich ośrodków, które mają artykuł), `N` (nagłówek grupy dyskusyjnej), `D` (nagłówek dystrybucji), `H` (wszystkie nagłówki), `o` (dane poglądowe) i `R` (dane replikacyjne).

Pole `parametry` jest specjalnie kodowane w zależności od sposobu dostarczania grup innym serwerom. W większości popularnych konfiguracji podajesz nazwę pliku wynikowego, do którego będziesz zapisywał wychodzące artykuły. W innych możesz go nie podawać. W jeszcze innych konfiguracjach ma ono różne znaczenia. Jeżeli chcesz zrobić coś niezwykłego, podręcznik *newsfeeds(5)* wyjaśni ci szczegółowo zastosowanie pola `parametry`.

Istnieje szczególna nazwa ośrodka, kodowana jako `ME`, którą powinien zawierać pierwszy wpis w pliku. Wpis ten jest używany do kontrolowania domyślnych ustawień dostarczania grup. Jeżeli wpis `ME` posiada związaną z dostarczaniem listę dystrybucji, będzie ona doklejana do każdego wpisu zawierającego ośrodek przed wysłaniem do niego grup. Pozwala to na przykład na automatyczne przekazywanie pewnych grup lub automatyczne blokowanie przekazania innych bez potrzeby powtarzania wzorca w każdym opisie ośrodka.

Wspomnieliśmy wcześniej, że możliwe jest użycie pewnych połączeń specjalnych do wygenerowania wątku danych, który ułatwia pracę przeglądarki. Wątek danych jest generowany za pomocą polecenia *overchan* będącego częścią dystrybucji INN. Wcześniej jednak trzeba stworzyć specjalną lokalną porcję o nazwie `overview`, przekazującą artykuły do polecenia *overchan* w celu przetworzenia na dane poglądowe.

Nasz serwer będzie udostępniał grupy tylko jednemu serwerowi zewnętrznemu, który znajduje się na uniwersytecie Groucho Marx i pobiera artykuły ze wszystkich grup, poza *control*, *junk* grupą lokalną *rec.crafts.brewing.private* i *rec.crafts.brewing.poison*, z której nie chcemy udostępniać artykułów wysłanych przez osoby z naszego browaru.

Do przesyłania grup przez NNTP do serwera **news.groucho.edu**, użyjemy polecenia *nntpsend*. Wymaga ono użycia metody dostarczania „file” i zapisywania ścieżki i ID artykułu. Zauważ, że ustawiliśmy pole `parametry` na nazwę pliku wynikowego. Nieco więcej o poleceniu *nntpsend* powiemy za chwilę. Nasza docelowa konfiguracja wygląda tak:

```
# Plik /etc/news/newsfeeds dla browaru wirtualnego
#
# Domy□lnie wysy□amy wszystkie grupy poza control i junk
ME:!control,!junk::
#
# Generujemy dane pogl□dowe dla przegl□darek grup.
overview::Tc,WO:/usr/lib/news/bin/overchan
#
# Dostarczamy uniwersytetowi Groucho Marx wszystko poza nasz□
# prywatn□ grup□ i artyku□ami wys□anymi na rec.crafts.brewing.
# poison,@rec.crafts.brewing.private:\
   Tf,Wnm:news.groucho.edu
#
```

Plik nntpsend.ctl

Program *nntpsend* zarządza przesyłaniem artykułów przez NNTP, wywołując polecenie *innxmit*. Widzieliśmy wcześniej proste zastosowanie polecenia *nntpsend*, ale ma ono też plik konfiguracyjny dający pewną elastyczność w konfigurowaniu dostarczania przez nas grup.

Polecenie *nntpsend* spodziewa się plików wsadowych dla ośrodków, do których ma wysyłać grupy. Oczekuje, że będą one miały nazwy postaci: */var/spool/news/out.going/*`nazwao□rodka`. *innd* tworzy te pliki wsadowe na podstawie wpisu w pliku *newsfeeds*, który widzieliśmy w poprzednim podrozdziale. W polu `parametry` określiliśmy nazwę ośrodka jako nazwę pliku i tym samym spełniliśmy wymagania co do danych wejściowych dla polecenia *nntpsend*.

Polecenie *nntpsend* ma plik konfiguracyjny o nazwie *nntpsend.ctl*, który zwykle znajduje się w katalogu */etc/news/*.

Plik *nntpsend.ctl* pozwala nam związać pełną nazwę domenową, ograniczenia rozmiaru przesyłanej porcji artykułów i ograniczenia parametrów transmisji z nazwą ośrodka. Nazwa ta ma unikalnie identyfikować logiczną, wysyłaną porcję artykułów. Ogólny format pliku jest następujący:

```
nazwao□rodka:fqdn:max_rozmiar:[argumenty]
```

Poniższa lista opisuje poszczególne elementy tego wiersza:

`nazwao□rodka`
: Nazwa ośrodka zgodnie z podaną w pliku *newsfeeds*.

`fqdn`
: Pełna nazwa domenowa serwera grup, do którego przesyłamy artykuły.

`max_rozmiar`
: Maksymalna wielkość porcji artykułów wysyłanych za jednym razem.

`argumenty`
: Dodatkowe argumenty przekazywane do polecenia *innxmit*.

Naszej przykładowej konfiguracji wystarczy bardzo prosty plik *nntpsend.ctl*. Mamy tylko jedno miejsce, do którego przekazujemy grupy. Ograniczymy jedną porcję do 2 MB i przekażemy poleceniu *innxmit* argument ustawiający czas oczekiwania na 3 minuty (180 sekund). Gdybyśmy byli większym ośrodkiem i mieli więcej porcji

grup, stworzylibyśmy podobne wpisy dla każdego ośrodka, do którego przekazywalibyśmy grupy.

```
# /etc/news/nntpsend.ctl
#
gmarxu:news.groucho.edu:2m:-t 180
#
```

Kontrolowanie dostępu przeglądarki

Jeszcze nie tak dawno temu różne organizacje bardzo chętnie udostępniały wszystkim serwery grup dyskusyjnych. Obecnie trudno jest znaleźć serwery publiczne. Większość organizacji skrupulatnie nadzoruje dostęp do swoich serwerów, zwłaszcza ogranicza dostęp użytkownikom swojej sieci. INN posiada pliki konfiguracyjne pozwalające na kontrolę dostępu.

Plik incoming.conf

We wprowadzeniu do INN-a wspomnieliśmy, że działa ono efektywnie i jest niewielkie dzięki rozdzieleniu mechanizmu przekazywania wiadomości innym serwerom od mechanizmu ich przeglądania. W pliku */etc/news/incoming.conf* umieszczasz hosty, które będą ci dostarczały grupy przez protokół NNTP, oraz pewne parametry sterujące sposobem, w jaki twój host pobiera z nich artykuły. Wszelkie hosty nie wpisane w tym pliku, a łączące się z gniazdem news nie będą obsługiwane przez demona *innd*, ale przez *nnrpd*.

Składnia pliku */etc/news/incoming.conf* jest bardzo prosta, ale jej zrozumienie może zająć chwilę. Dopuszczalne są trzy typy wpisów: para klucz/wartość, która określa sposób podawania atrybutów i ich wartości parametry równorzędne, które określają sposób podawania nazw hosta, który może wysyłać do nas artykuły przez NNTP oraz grupy, których dotyczą poprzednie wartości. Pary klucz/wartość mogą mieć trzy różne zakresy. Pary globalne dotyczą każdego elementu zdefiniowanego w pliku, grupy par dotyczą wszystkich elementów zdefiniowanych w danej grupie, a pary równoważne dotyczą tylko danego konkretnego przypadku. Definicje bardziej szczegółowe unieważniają te mniej szczegółowe i dlatego definicje równoważne unieważniają definicje grup, które z kolei unieważniają pary globalne.

Nawiasy klamrowe ({}) są używane do oznaczenia początku i końca definicji `group` i `peer`. Znak # oznacza, że dalszy ciąg wiersza to komentarz. Pary klucz/wartość są oddzielane dwukropkiem i są wpisywane w wierszu pojedynczo.

Można podać szereg różnych kluczy. Najczęściej używane i najbardziej przydatne z nich to:

hostname

Ten klucz określa, oddzielaną przecinkami, listę pełnych nazw domenowych lub adresów IP hostów równorzędnych, które mogą wysyłać nam artykuły. Jeżeli ten klucz nie zostanie podany, przyjmowana jest domyślna nazwa hosta partnerskiego.

streaming
: Ten klucz określa, czy dla danego hosta są dopuszczalne polecenia strumieniowe. Jest to wartość boole'owska, domyślnie – `true`.

max-connections
: Ten klucz określa maksymalną liczbę połączeń dopuszczalnych z danej grupy lub z hostów równoważnych. Wartość zero oznacza nieograniczoną ich liczbę (można także podać `none`).

password
: Ten klucz pozwala ci określić hasło, które musi być używane przez partnera, jeżeli ma on prawo przesyłać wiadomości. Domyślnie hasło nie jest wymagane.

patterns
: Ten klucz określa grupy, które przyjmujemy od partnera. Pole to jest kodowane zgodnie z tymi samymi regułami, których używaliśmy w pliku *newsfeeds*.

W naszym przykładzie mamy tylko jeden host, który może nam dostarczać grupy: nasz dostawca z uniwersytetu Groucho Marx. Nie potrzebujemy hasła, ale nie będziemy przyjmować z zewnątrz żadnych artykułów do naszych prywatnych grup. Nasz *hosts.nntp* wygląda tak:

```
# Plik incoming.conf browaru wirtualnego

# Ustawienia globalne
streaming:       true
max-connections:    5

# Pozwalamy na wysy□anie NNTP z naszego hosta lokalnego
peer ME {
   hostname: "localhost, 127.0.0.1"
}

# Pozwalamy groucho na wysy□anie nam wszystkich grup poza lokalnymi.
peer groucho {
   hostname: news.groucho.edu
   patterns: !rec.crafts.brewing.private
}
```

Plik nnrp.access

Wspomnieliśmy już, że przeglądarki grup, a w rzeczywistości wszelkie hosty nie uwzględnione w pliku *hosts.nntp*, które łączą się z serwerem grup INN, są obsługiwane przez program *nnrpd*. Program ten używa pliku */etc/news/nnrp.access* do określenia, kto ma prawo korzystać z serwera grup i jakie powinien mieć prawa dostępu.

Plik *nnrp.access* ma budowę podobną do innych plików konfiguracyjnych, które omawialiśmy do tej pory. Składa się z zestawu wzorców używanych do dopasowywania nazw lub adresów IP łączących się hostów i pól, które określają, jakie prawa dostępu powinny być im dane. Każdy wpis powinien znajdować się w oddzielnym wierszu, a pola powinny być oddzielone dwukropkami. Jako zwykle używany będzie ostatni wpis w pliku pasujący do podłączającego się hosta, a więc powinieneś

umieszczać wzorce ogólne na początku, a następnie wzorce szczegółowe. Pięć pól w każdym wpisie ma następujące znaczenie:

Nazwa hosta lub adres IP
: To pole jest zgodne z regułami dopasowania wzorca *wildmat(3)*. Jest to wzorzec opisujący nazwę hosta lub adres IP podłączającego się hosta.

Prawa dostępu
: To pole określa, jakie prawa dostępu powinny być nadane pasującemu hostowi. Istnieją dwa rodzaje praw, które możesz skonfigurować: `R` daje prawo czytania, a `P` daje prawo wysyłania.

Nazwa użytkownika
: To pole jest opcjonalne i pozwala określić nazwę użytkownika, na które go musi się zalogować klient NNTP, zanim będzie mógł wysyłać artykuły. Pole to można pozostawić puste. Do czytania artykułów nie jest potrzebna żadna autoryzacja.

Hasło
: To pole jest opcjonalne i zawiera hasło towarzyszące polu `nazwa użytkownika`. Pozostawienie tego pola pustego oznacza, że do wysyłania artykułów nie jest wymagane hasło.

Grupy
: To pole jest wzorcem określającym, do których grup klient ma dostęp. Wzorzec podlega tym samym regułom, jakie były używane w pliku *newsfeeds*. Domyślną wartością tego pola jest brak grup, a więc zwykle podajesz tu jakiś wzorzec.

W przykładzie browaru wirtualnego pozwalamy każdemu klientowi NNTP z domeny browaru na czytanie wszystkich grup i wysyłanie do nich. Wszystkim innym klientom NNTP dajemy prawo do czytania wszystkich grup poza naszymi wewnętrznymi grupami prywatnymi. Nasz plik *nnrp.access* będzie wyglądał następująco:

```
# Virtual Brewery - nnrp.access
# Pozwalamy publicznie czytać wszystkie grupy poza prywatnymi.
*:R:::*,!rec.crafts.brewing.private

# Każdy host z domeny browaru może czytać i wysyłać artykuły
# do wszystkich grup
*.vbrew.com:RP::*
```

Wygasanie artykułów w grupach

Artykuły odbierane przez serwer grup są zapisywane na dysk. Aby to miało sens, artykuły muszą być dostępne dla użytkowników przez jakiś okres czasu, a więc duży serwer grup może zajmować wiele miejsca na dysku. Aby miejsce to było wykorzystywane efektywnie, możesz włączyć automatyczne usuwanie artykułów po zadanym okresie czasu. Nazywa się to *wygaśnięciem artykułu*. Oczywiście INN obsługuje automatyczne wygasanie artykułów.

Plik expire.ctl

Do usuwania starych artykułów serwer INN używa programu *expire*. Program ten z kolei wykorzystuje plik */etc/news/expire.ctl*, w którym są skonfigurowane reguły zarządzające wygasaniem artykułów.

Składnia pliku */etc/news/expire.ctl* jest dość prosta. Podobnie jak w większości plików konfiguracyjnych, puste wiersze lub wiersze rozpoczynające się znakiem # są ignorowane. Ogólna zasada jest taka, że każdą regułę piszesz w oddzielnym wierszu. Każda reguła definiuje, jak jest realizowane wygasanie artykułu w grupach zgodnych z zadanym wzorcem. Składnia reguły wygląda tak:

```
wzorzec:znmod:trzymanie:domy□lnie:czyszczenie
```

Poniższa lista opisuje poszczególne pola reguły:

wzorzec
: To pole jest oddzieloną przecinkami listą wzorców do pasowujących nazwy grup. Do ich sprawdzania jest używana procedura *wildmat(3)*. Stosowana jest ostatnia z pasujących reguł, a więc gdybyś chciał umieszczać reguły ze stawień uniwersalnych (*), powinny być w pliku jako pierwsze.

znmod
: Ten znacznik opisuje, jak reguła jest stosowana do grup moderowanych. Może on być zapisany jako `M`, co oznacza, że reguła dotyczy tylko grup moderowanych, albo jako `U`, co oznacza, że reguła dotyczy tylko grup niemoderowanych. Można też użyć `A`, aby wskazać, że reguła ignoruje status moderowania i dotyczy wszystkich grup.

trzymanie
: To pole pozwala określić minimalny czas, przez jaki będzie przechowywany artykuł z ustawionym polem `Expires` w nagłówku, zanim wygaśnie. Wartość jest określana w dniach, ale dopuszczalne są liczby zmiennoprzecinkowe, a więc możesz podać wartość `7.5`, która oznacza siedem i pół dnia. Możesz także podać `never`, jeżeli chcesz, by artykuł pozostał w grupie na zawsze.

domyślnie
: To pole jest najważniejsze. Pozwala określić czas, przez jaki będzie przechowywany artykuł bez pola nagłówka `Expires`. Większość artykułów nie będzie miała takiego pola w nagłówku. Pole *`domy□lnie`* jest kodowane dokładnie w ten sam sposób jak pole *`trzymanie`*. `never` oznacza, że artykuł bez nagłówka `Expires` nigdy nie wygaśnie.

czyszczenie
: To pole pozwala zadać maksymalny czas, przez jaki będzie przechowywany artykuł z polem `Expires`, zanim wygaśnie. Kodowanie tego pola przebiega tak samo jak pola `trzymanie`.

Nasze wymagania są proste. Będziemy przechowywać wszystkie artykuły we wszystkich grupach domyślnie przez 14 dni, natomiast artykuły z nagłówkiem `Expires`, od 7 do 21 dni. Grupa *rec.crafts.brewing.private* jest naszą grupą wewnętrzną, a więc nie chcemy, by jakiekolwiek artykuły w niej zaważ wygasały:

```
# plik expire.ctl dla browaru wirtualnego

# Domyślne wygasanie wszystkich artykułów po 14 dniach. Od 7
# do 21 dni dla tych, które mają nagłówek Expires:
*:A:7:14:21

# To jest specjalna grupa wewnętrzna, która nie wygasa.
rec.crafts.brewing.private:A:never:never:never
```

Powiemy jeszcze o jednym specjalnym rodzaju wpisu, który może się znaleźć w twoim pliku */etc/news/expire.ctl*. Możesz mieć dokładnie jeden wiersz wyglądający tak:

```
/remember/:dni
```

Pozwala on zadać minimalną liczbę dni, przez jaką artykuł będzie pamiętany w pliku historii, bez względu na to, czy sam artykuł wygasł, czy nie. Może to być przydatne, jeżeli jeden z ośrodków dostarczających nam artykuły nie robi tego często i ma tendencję do przysyłania starych artykułów. Ustawienie pola `/remember/` pomaga zapobiec ponownemu przysyłaniu artykułu, który u nas już wygasł. Jeżeli twój serwer pamięta, że już otrzymał kiedyś taki artykuł, odrzuci próbę ponownego jego przysłania. Trzeba pamiętać, że to ustawienie nie ma żadnego wpływu na wygasanie artykułu. Dotyczy jedynie czasu przechowywania informacji o artykule w pliku historii.

Obsługiwanie wiadomości kontrolnych

Tak jak C News, tak i INN może automatycznie przetwarzać wiadomości kontrolne. INN ma doskonały mechanizm konfiguracyjny nadzorujący działania, jakie są podejmowane dla każdej z wiadomości kontrolnych, oraz mechanizm kontroli dostępu, który czuwa nad tym, kto zainicjował działanie i wobec jakich grup.

Plik control.ctl

Budowa pliku *control.ctl* jest dosyć prosta. Reguły składniowe są w zasadzie takie same jak w przypadku innych plików konfiguracyjnych INN-a. Wiersze rozpoczynające się od znaku # są ignorowane; wiersze można kontynuować używając znaku /, a pola są oddzielane dwukropkami.

Gdy zostanie odebrana wiadomość kontrolna, jest sprawdzana po kolei z każdą regułą. Stosowana jest jakzwykle ostatnia pasująca reguła w pliku, a więc powinieneś umieszczać ogólne reguły na początku pliku, a dokładniejsze pod jego koniec. Ogólna składnia pliku jest następująca:

```
wiadomość:skąd:grupa:działanie
```

Znaczenie poszczególnych pól jest następujące:

wiadomość
: Jest to nazwa wiadomości kontrolnej. Typowe wiadomości kontrolne opisujemy dalej.

skąd
: Jest to wzorzec opisujący adres e-mail osoby wysyłającej wiadomość. Przed porównaniem adres jest konwertowany do małych liter.

grupa
: Jeżeli wiadomość kontrolna to `newgroup` lub `rmgroup`, to pole ma postać wzorca pasującego do tworzonej lub usuwanej grupy.

działanie
: To pole określa, jakie działanie podjąć, gdy wiadomość pasuje do reguły. Możliwe są różne działania, opisane na następnej liście.

Pole `wiadomo□□` w każdym wierszu może przyjmować następujące wartość:

checkgroups
: Ta wiadomość stanowi żądanie wobec administratora grup, by zsynchronizował swoją bazę aktywnych grup z listą grup dostarczoną w wiadomości kontrolnej.

newgroup
: Ta wiadomość stanowi żądanie utworzenia nowej grupy. Treść wiadomości powinna zawierać krótki opis przeznaczenia tworzonej grupy.

rmgroup
: Ta wiadomość stanowi żądanie usunięcia grupy.

sendsys
: Ta wiadomość stanowi żądanie przesłania pocztą pliku *sys* z serwera grup do nadawcy wiadomości. RFC-1036 mówi, że warunkiem członkostwa w Usenecie jest publiczne udostępnianie tej wiadomości, ponieważ jest ona wykorzystywana przy uaktualnianiu map usenetowych.

version
: Ta wiadomość stanowi żądanie zwrotu do nadawcy tej wiadomości nazwy hosta i wersji oprogramowania serwera grup.

all
: Jest to specjalny zapis, do którego pasują wszystkie wiadomości kontrolne.

Pole `wiadomo□□` może zawierać następujące działania:

doit
: Żądane polecenie jest wykonywane. W wielu przypadkach do administratora jest wysyłana wiadomość e-mail z informacją, że dane działanie miało miejsce.

doit=`plik`
: Jest to działanie identyczne z `doit`, ale do pliku log*plik* zostanie zapisany komunikat. Jeżeli podanym plikiem jest *mail*, wpis log jest wysyłany pocztą. Jeżeli podanym plikiem jest ciąg pusty, wiadomość log jest zapisywana do */dev/null* i jest to równoważne z użyciem czystego polecenia `doit`. Jeżeli nazwa *plik* rozpoczyna się od znaku /, jest uznawana za bezwzględną ścieżkę do pliku log. W przeciwnym razie zadana nazwa jest zamieniana do postaci */var/log/news/file.log*.

doifarg
: Żądane polecenie jest wykonywane, jeżeli ma argument. Jeżeli nie ma argumentu, wiadomość kontrolna jest ignorowana.

drop
: Żądane polecenie jest ignorowane.

log
: Wiadomość log jest wysyłana na standardowe wyjście błędów `stderr` procesu *innd*. Normalnie jest przekierowywana do pliku */var/log/news/errlog*.

log=`plik`
: To samo co `log`, ale plik log jest zadawany na tych samych zasadach co w przypadku działania `doit=plik`.

mail
: Do administratora jest wysyłana wiadomość e-mail zawierająca żądane szczegóły. Żadne inne działanie nie jest podejmowane.

*verify-**
: Jeżeli działanie rozpoczyna się od ciągu "`verify-`", wiadomość kontrolna jest uwierzytelniana przez PGP (lub GPG)*.

Abyś mógł zobaczyć, jak plik *control.ctl* wygląda w rzeczywistości, pokazujemy prosty przykład:

```
## Przyk□adowy plik /etc/news/control.ctl
##
## Uwaga: nie powiniene□ u□ywa□ tego pliku - s□u□y on tylko do
## demonstracji

##   Obs□uga wiadomo□ci kontrolnych
all:*:*:mail
checkgroups:*:*:mail
ihave:*:*:drop
sendme:*:*:drop
sendsys:*:*:log=sendsys
senduuname:*:*:log=senduuname
version:*:*:log=version
newgroup:*:*:mail
rmgroup:*:*:mail

## Obs□uga wiadomo□ci kontrolnych dla o□miu najwa□niejszych
## hierarchii grup
## COMP, HUMANITIES, MISC, NEWS, REC, SCI, SOC, TALK
checkgroups:*:comp:*|humanities.*|misc.*|news.*|rec.*|sci.*|talk.*:drop
newgroup:*:comp:*|humanities.*|misc.*|news.*|rec.*|sci.*|talk.*:drop
rmgroup:*:comp:*|humanities.*|misc.*|news.*|rec.*|sci.*|talk.*:drop
checkgroups:group-admin@isc.org:*:verify-news.announce.newgroups
newgroup:group-admin@isc.org:comp.*|misc.*|news.*:verify-news.announ-
ce.newgroups
newgroup:group-admin@isc.org:rec.*|sci.*|soc.*:verify-news.announce.newgroups
newgroup:group-admin@isc.org:talk.*|humanities.*:verify-news.announce.newgroups
```

* PGP i GPG to narzędzia stworzone do uwierzytelniania lub szyfrowania wiadomości za pomocą technik klucza publicznego. GPG jest wersją GNU PGP. GPG można znaleźć pod adresem *http://www/gnupg.org/*, a PGP pod adresem *http://www.pgp.com/*.

```
rmgroup:group-admin@isc.org:comp.*|misc.*|news.*:verify-news.announce.newgroups
rmgroup:group-admin@isc.org:rec.*|sci.*|soc.*:verify-news.announce.newgroups
rmgroup:group-admin@isc.org:talk.*|humanities.*:verify-news.announce.newgroups

## GNU ( Free Software Foundation )
newgroup:gnu@prep.ai.mit.edu:gnu.*:doit
newgroup:news@*ai.mit.edu:gnu.*:doit
rmgroup:gnu@prep.ai.mit.edu:gnu.*:doit
rmgroup:news@*ai.mit.edu:gnu.*:doit

## LINUX (Newsfeed from news.lameter.com)
checkgroups:chrisoph@lameter.com:linux.*:doit
newgroup:chrisoph@lameter.com:linux.*:doit
rmgroup:christoph@lameter.com:linux.*:doit
```

Eksploatowanie INN-a

Pakiet źródłowy INN zawiera skrypt uruchamiający *inn* w czasie inicjacji systemu. Skrypt zwykle nosi nazwę */usr/lib/news/bin/rc.news*. Czyta on argumenty z innego skryptu o nazwie */usr/lib/news/innshellvars*, który zawiera nazwy plików i ścieżki używane przez *inn* do lokalizacji potrzebnych mu elementów. Dobrze jest uruchamiać *inn* z prawami dostępu użytkownika innego niż root, na przykład `news`.

Aby *inn* na pewno uruchomił się w czasie w czasie startu systemu, powinieneś sprawdzić, czy plik */usr/lib/news/innshellvars* jest skonfigurowany poprawnie, a następnie wywołać skrypt */usr/lib/news/bin/rc.news* ze skryptu uruchamianego w czasie startu.

Ponadto, co jakiś czas należy wykonywać pewne zadania administracyjne. Zwykle konfiguruje się je tak, aby były uruchamiane poleceniem *cron*. Najlepszym sposobem na zrobienie tego jest dodanie odpowiednich poleceń do pliku */etc/crontab* lub stworzenie pliku odpowiedniego dla katalogu */etc/cron.d*, jeżeli twoja dystrybucja to obsługuje. Taki przykład owy plik może wyglądać następująco:

```
# Przyk□adowy plik /etc/cron.d/inn u□ywany w dystrybucji Debian
#
SHELL=/bin/sh
PATH=/usr/lib/news/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Wyga□ni□cie starych artyku□ów i wygenerowanie raportów
# ka□dej nocy

15 0 * * * news news.daily expireover lowmark delayrm

# Co godzin□ uruchomienie rnews -U. Nie jest to tylko dla
# o□rodków UUCP, ale tak□e przetwarza artyku□y skolejkowane
# przez in.nnrpd w sytuacji, gdy innd nie przyjmowa□ □adnych
# artyku□ów.

10 * * * * news rnews -U
```

Codziennie te polecenia automatycznie usuwają stare artykuły oraz co godzinę przetwarzają artykuły z kolejki. Zauważ, że są uruchamiane z prawami użytkownika `news`.

Zarządzanie INN-em: polecenie ctlinnd

Serwer grup INN zawiera polecenie do zarządzania jego codziennym działaniem. Polecenie *ctlinnd* może być używane do operowania na grupach i porcjach grup wysyłanych do innych serwerów, uzyskiwania stanu serwera oraz do przeładowywania, zatrzymywania i uruchamiania serwera.

Podsumowanie działania polecenia *ctlinnd* możesz uzyskać, używając polecenia następująco:

```
# ctlinnd -h
```

Omówimy tu kilka z ważniejszych zastosowań *ctlinnd*. Więcej szczegółów znajdziesz na stronie podręcznika poświęconej temu poleceniu.

Dodawanie nowej grupy

Aby dodać nową grupę, użyj polecenia następująco:

`ctlinnd newgroup` *`grupa znacz twórca`*

Argumenty mają następujące znaczenie:

`grupa`
: Nazwa tworzonej grupy.

`znacz`
: Ten argument powinien być zapisywany w ten sam sposób jak pole *`znaczniki`* pliku *active*. Domyślnie jest przyjmowana wartość `y`, jeżeli nic innego nie zostanie podane.

`twórca`
: Nazwa osoby tworzącej grupę. Umieść ją w cudzysłowie, jeżeli chcesz wpisać jakieś spacje.

Zmiana grupy

Aby zmienić grupę, użyj polecenia następująco:

`ctlinnd changegroup` *`grupa znacz`*

Argumenty mają następujące znaczenie:

`grupa`
: Nazwa zmienianej grupy.

`znacz`
: Ten argument powinien być zapisywany w ten sam sposób co pole *`znaczniki`* pliku *active*.

To polecenie przydaje się przy zmianie statusu moderowania grupy.

Usuwanie grupy

Aby usunąć grupę, użyj polecenia następująco:

```
ctlinnd rmgroup grupa
```

Argument ma następujące znaczenie:

grupa
: Nazwa usuwanej grupy.

To polecenie usuwa zadaną grupę z pliku *active*. Nie ma wpływu na katalog bufora. Wszystkie zawarte w nim artykuły wygasną w zwykły sposób, a nowe nie będą przyjmowane.

Przenumerowanie grupy

Aby przenumerować grupę, użyj polecenia następująco:

```
ctlinnd renumber grupa
```

Argument ma następujące znaczenie:

grupa
: Nazwa przenumerowanej grupy. Jeżeli *grupa* jest pustym ciągiem znaków, przenumerowywane są wszystkie grupy.

To polecenie uaktualnia dolny znacznik w zadanej grupie.

Używanie serwera przez przeglądarki grup – pozwalanie i zabranianie

Użyj poniższego polecenia, aby pozwolić lub zabronić przeglądarkom korzystać z serwera:

```
ctlinnd readers znacz tekst
```

Argumenty mają następujące znaczenie:

znacz
: Jeśli jest ustawiony na `n`, zabrania podłączać się przeglądarkom grup. Podanie `y` pozwala na przyjmowanie połączeń od przeglądarek.

tekst
: Podany tekst jest przekazywany do przeglądarki, która próbuje się podłączyć i zwykle opisuje powód zabronienia dostępu. Przy ponownym włączaniu dostępu przeglądarkom, pole to musi być pustym ciągiem lub kopią tekstu podanego przy zakazie dostępu.

To polecenie nie wpływa na przychodzące z innych serwerów porcje grup. Kontroluje jedynie dostęp przeglądarek.

Odmowa połączenia z innego serwera

Aby odmówić połączenia innemu serwerowi, użyj polecenia następująco:

```
ctlinnd reject powód
```

Argument ma następujące znaczenie:

powód
: Podany tekst powinien wyjaśniać, dlaczego przychodzące do *innd* połączenia są odrzucane.

To polecenie nie ma wpływu na połączenia obsługiwane przez *nnrpd* (tzn. przeglądarki grup). Dotyczy jedynie połączeń, które są obsługiwane bezpośrednio przez *innd*, czyli połączeń z innych serwerów.

Pozwolenie na połączenia z innego serwera

Aby pozwolić na połączenia innemu serwerowi, użyj polecenia następująco:

```
ctlinnd allow powód
```

Argument ma następujące znaczenie:

powód
: Podany tekst musi być identyczny z podanym w poleceniu *reject* lub musi być to ciąg pusty.

To polecenie odwraca działanie polecenia *reject*.

Zamknięcie serwera grup

Aby zamknąć serwer grup, użyj polecenia następująco:

```
ctlinnd throttle powód
```

Argument ma następujące znaczenie:

powód
: Powód zamknięcia serwera.

To polecenie jest równoważne z jednoczesnym wydaniem polecenia `newsreaders no` i `reject`. Jest przydatne przy pracach awaryjnych wykonywanych na bazie serwera grup. Daje pewność, że nikt nie będzie próbował go uaktualnić, gdy przy nim pracujesz.

Restart serwera grup

Aby zrestartować serwer grup, użyj polecenia następująco:

```
ctlinnd go powód
```

Argument ma następujące znaczenie:

powód
: Powód zatrzymania serwera. Jeżeli pole to jest pustym ciągiem znaków, serwer zostanie bezwarunkowo ponownie włączony. Jeżeli powód został podany, to tylko funkcje, które są wyłączone z powodu zgodnego z podanym tekstem, zostaną ponownie uruchomione.

To polecenie jest używane do ponownego uruchamiania serwera po wykonaniu poleceń *throttle*, *pause* lub *reject*.

Wyświetlanie statusu pobierania plików z innego serwera

Aby wyświetlić status pobierania plików z innego serwera, użyj polecenia następująco:

```
ctlinnd feedinfo ośrodek
```

Argument ma następujące znaczenie:

ośrodek
: Nazwa ośrodka (wzięta z pliku *newsfeeds*), dla którego chcesz wyświetlić status.

Odłączenie dostarczania plików z innego serwera

Aby wyłączyć dostarczanie plików z innego serwera, użyj polecenia następująco:

```
ctlinnd drop ośrodek
```

Argument ma następujące znaczenie:

ośrodek
: Nazwa ośrodka (wzięta z pliku *newsfeeds*), dla którego dostarczanie plików jest wyłączane. Jeżeli pole jest pustym ciągiem, wszystkie aktywne transmisje zostaną przerwane.

Wyłączenie dostarczania plików za trzymuje wszelkie aktywne transmisje do danego ośrodka. Nie jest to zmiana stała. Polecenie jest przydatne, jeżeli modyfikujesz szczegóły związane z przesyłaniem plików z danego ośrodka, a transmisja jest akurat aktywna.

Rozpoczynanie dostarczania plików z innego serwera

Aby rozpocząć dostarczanie plików z innego serwera, użyj polecenia następująco:

```
ctlinnd begin ośrodek
```

Argument ma następujące znaczenie:

ośrodek
: Nazwa ośrodka wzięta z pliku *newsfeeds*, z którego rozpoczęto przesyłanie plików. Jeżeli przesyłanie z tego ośrodka jest już aktywne, automatycznie jest wykonywane polecenie *drop*.

To polecenie powoduje, że serwer czyta ponownie plik *newsfeeds*, znajduje pasujący wpis i rozpoczyna przesyłanie plików do/z danego ośrodka zgodnie ze znalezionymi szczegółami. Możesz użyć tego polecenia do przetestowania nowych wpisów po ich dodaniu lub modyfikacji w pliku *newsfeeds*.

Anulowanie artykułu

Aby anulować artykuł, użyj polecenia następująco:

```
ctlinnd cancel ID-artykułu
```

Argument ma następujące znaczenie:

ID-artykułu
: ID anulowanego artykułu.

To polecenie powoduje, że zadany artykuł zostanie usunięty z serwera. Nie generuje wiadomości cancel.

24

Konfigurowanie przeglądarki grup dyskusyjnych



Przeglądarka grup to program, który użytkownik uruchamia do oglądania, zachowywania i tworzenia artykułów w grupach dyskusyjnych. Do Linuksa przeniesiono kilka przeglądarek grup. Opiszemy podstawową konfigurację trzech najpopularniejszych: *tin*, *trn* i *nn*.

Jedną z najbardziej efektywnych przeglądarek jest następujące polecenie:

```
$ find /var/spool/news -name '[0-9]*' -exec cat {} \; | more
```

W ten sposób artykuły z grup czytają uniksowi twardziele.

Większość przeglądarek jest jednak bardziej wyrafinowana. Zwykle mają pełnoekranowy interfejs z oddzielnymi poziomami do wyświetlania wszystkich grup, do których użytkownik się zapisał, do przeglądania listy artykułów w każdej grupie i pojedynczych artykułów. Wiele przeglądarek WWW działa również jako przeglądarki grup, ale jeżeli chcesz używać niezależnej przeglądarki – grup, ten rozdział wyjaśnia, jak skonfigurować dwie podstawowe: *trn* i *nn*.

Na poziomie grup większość przeglądarek wyświetla listę artykułów, pokazując wiersz z tytułem i autorem. W dużych grupach trudno jest śledzić artykuły ze sobą związane, choć możliwe jest identyfikowanie odpowiedzi na wcześniejsze artykuły.

Odpowiedzi zwykle noszą tytuł oryginalnego artykułu z przedrostkiem `Re:`. Ponadto wiersz nagłówka `Reference:` powinien zawierać ID wiadomości, na którą artykuł stanowi odpowiedź. Sortowanie artykułów według tych dwóch kryteriów daje niewielkie zestawy artykułów (w rzeczywistości drzewa), które są nazywane *wątkami*. Jednym z zadań przy tworzeniu przeglądarki grup jest wynalezienie efektywnego sposobu obsługi wątków, ponieważ czas do tego potrzebny jest proporcjonalny do kwadratu liczby artykułów.

Nie będziemy wnikać w to, jak są zbudowane interfejsy użytkownika. Wszystkie obecnie dostępne dla Linuksa przeglądarki mają doskonałą funkcję pomocy, a więc skorzystaj z niej, jeżeli chcesz poznać więcej szczegółów.

W kolejnych podrozdziałach zajmiemy się jedynie zadaniami administracyjnymi. Większość z nich ma związek z tworzeniem baz wątków i liczeniem.

Konfigurowanie tina

Najbardziej wszechstronną przeglądarką obsługującą wątki jest *tin*. Została ona napisana przez Iaina Lea i na wiązuje do starszej przeglądarki *tass* (napisanej przez Richa Skrenta). Podział na wątki odbywa się w momencie wejścia przez użytkownika do grupy i jest naprawdę szybki, pod warunkiem, że nie korzystasz z NNTP.

Na komputerze 486DX50 podzielenie tysiąca artykułów na wątki zajmuje 30 sekund, jeśli są odczytywane bezpośrednio z dysku. Natomiast przy podłączeniu się do obciążonego serwera NNTP trwa to ponad 5 minut*. Możesz skrócić ten czas, regularnie uaktualniając plik indeksu przez wywołanie *tina* z opcją *–u*, tak że gdy następnym razem uruchomisz *tina*, wątki już będą istniały. Możesz także wywołać *tina* z opcją *–U* przy czytaniu grup. Przy takim wywołaniu *tin* tworzy działający w tle proces, który tworzy pliki indeksów, kiedy ty czytasz grupy.

Zwykle *tin* zapisuje bazy wątków w katalogu macierzystym użytkownika w podkatalogu *.tin/index*. Może to pochłaniać dużo zasobów, a więc chyba lepiej mieć jedną bazę w centralnym miejscu. W tym celu należy ustawić dla *tina* na przykład prawo set uid **news**. *tin* będzie w takiej sytuacji przechowywał bazy wątków w katalogu */var/spool/news/.index*. W przypadku dostępu do plików lub w wywołaniu powłoki będzie zmieniał efektywny uid na rzeczywisty uid wywołującego go użytkownika**.

Wersja *tina* dołączona do pewnych dystrybucji Linuksa jest skompilowana bez obsługi NNTP, ale większość ją ma. Gdy wywołasz *tina* jako *rtin* lub z opcją *–r*, próbuje on połączyć się z serwerem NNTP podanym w pliku */etc/nntpserver* lub w zmiennej środowiskowej `NNTPSERVER`. Plik *nntpserver* po prostu zawiera nazwę serwera umieszczoną w oddzielnym wierszu.

Konfigurowanie trn

trn również jest następcą starszej przeglądarki grup, *rn* (skrót od ang. *read news*). „t" w nazwie pochodzi od słowa *threaded* (obsługująca wątki). Została ona napisana przez Wayne'a Davidsona.

W odróżnieniu od *tina*, *trn* nie ma możliwości generowania bazy wątków w czasie pracy. Wykorzystuje za to wątki przygotowane przez program *mthreads*, który musi być regularnie wywoływany z *crona* w celu aktualizacji plików indeksu.

Możesz czytać nowe artykuły bez uruchomionego *mthreads*, ale wtedy stale będziesz napotykać porozrzucane tytuły, takie jak „PRAWDZIWA OKAZJA!", które *mthreads* umieściłby w jednym wątku, który łatwo pominąć.

---

* Po prawdzie się to znacznie, jeżeli serwer sam dokonuje podziału na wątki i przekazuje bazę wątków klientowi. Na przykład tak robi INN .

** Z tego powodu będziesz widział brzydkie komunikaty błędów przy wywoływaniu *tina* jako superużytkownik. W końcu nie powinieneś wykonywać rutynowych zadań jako **root**.

Aby włączyć wątki dla konkretnych grup, wywołaj *mthreads* z listą grup podaną w wierszu poleceń. Format listy jest taki sam jak w pliku *sys* w C News:

```
$ mthreads 'comp,rec,|rec.games.go'
```

To polecenie włącza wątki dla wszystkich grup *comp* i *rec*, za wyjątkiem *comp.games.go* (ludzie, którzy grają w go, nie potrzebują luksusowych wątków). Następnie możesz normalnie wywołać *mthreads* bez żadnych opcji, aby podzielić na wątki wszystkie nowo przychodzące artykuły. Podział na wątki wszystkich grup znajdujących się w twoim pliku *active* może być włączony przez wywołanie *mthreads* z listą grup *all*.

Jeżeli otrzymujesz artykuły z grup w nocy, zwykle będziesz uruchamiał *mthreads* rano, ale możesz także robić to częściej, jeżeli jest taka potrzeba. Duże, obciążone ośrodki zechcą uruchamiać *mthreads* w trybie demona. Gdy zostanie on uruchomiony w czasie inicjacji systemu z opcją *–d*, pracuje w tle i budzi się co dziesięć minut, by sprawdzić, czy nie nadeszły nowe artykuły, które trzeba podzielić na wątki. Aby uruchomić *mthreads* w trybie demona, umieść poniższy wiersz w swoim skrypcie *rc.news*:

```
/usr/local/bin/rn/mthreads -deav
```

Opcja *–a* powoduje, że *mthreads* automatycznie włącza dzielenie na wątki nowych grup, zaraz po ich utworzeniu, a *-v* włącza komunikaty umieszczane przez *mthreads* w pliku *mt.log* w katalogu, w którym jest zainstalowany *trn*.

Stare artykuły, które nie są już dostępne, muszą być regularnie usuwane z plików indeksowych. Domyślnie tylko artykuły o numerze niższym od dolnego znacznika będą usuwane*. Artykuły o numerach większych, które wygasły (ponieważ najstarszemu artykułowi został przypisany dłuższy czas wygaśnięcia przez pole nagłówka `Expires:`), mogą mimo wszystko zostać usunięte przez podanie opcji *–e* wymuszającej wygasanie „rozszerzone". Gdy *mthreads* działa w trybie demona, opcja *–e* powoduje, że *mthreads* wykonuje takie „rozszerzone" wygasanie raz dziennie, zaraz po północy.

Konfigurowanie nn

nn, napisana przez Kima F. Storma, zdaje się być przeglądarką, której głównym celem nie jest czytanie grup. Jej nazwa pochodzi od słów *No News* (brak wiadomości), a jej mottem są słowa „No news is good news, *nn* is better" (brak wiadomości to dobra wiadomość, *nn* jest lepsza).

Aby osiągnąć ten ambitny cel, *nn* posiada szereg narzędzi pomocniczych, które nie tylko pozwalają generować wątki, ale także intensywnie sprawdzać spójność bazy danych, liczyć i zbierać statystyki użytkowania oraz ograniczać dostęp. Istnieje też program administracyjny *nnadmin*, który pozwala na interaktywne wykonywanie tych zadań. Jest on bardzo intuicyjny, a więc nie będziemy się na nim skupiać. Omówimy jedynie generowanie plików indeksów.

* Zauważ, że C News (opisany w rozdziale 21, *C News*) nie uaktualnia automatycznie tego znacznika – musisz uruchamiać *updatemin*, aby to zrobić.

Menedżer bazy wątków *nn* nosi nazwę *nnmaster*. Zwykle jest uruchamiany jako demon w pliku *rc* w czasie startu komputera. Jest wywoływany tak:

```
/usr/local/lib/nn/nnmaster -l -r -C
```

To polecenie włącza podział na wątki dla wszystkich grup obecnych w pliku *active*.

Podobnie możesz wywoływać *nnmaster* okresowo z *crona*, podając listę grup, na których ma działać. Jest to bardzo podobne do listy subskrypcyjnej w pliku *sys*, z tą różnicą, że używa się spacji zamiast przecinków. Zamiast sztucznej grupy **all**, do oznaczenia wszystkich grup należy użyć argumentu pustego `""`. Przykładowe wywołanie wygląda tak:

```
# /usr/local/lib/nn/nnmaster !rec.games.go rec comp
```

Zauważ, że kolejność jest istotna. Zawsze wygrywa ta pasująca grupa, która znajduje się najbardziej po lewej stronie. Dlatego, gdybyśmy umieścili `!rec.games.go` po `rec`, wszystkie artykuły z tej grupy byłyby podzielone na wątki bez względu na wszystko.

nn oferuje kilka sposobów usuwania wygaśniętych artykułów z baz danych. Pierwszym jest uaktualnianie bazy przez przeglądanie katalogów grup i odrzucanie wpisów odnoszących się do artykułów, którym upłynął czas przechowywania. Jest to domyślne działanie uzyskiwane przez wywołanie *nnmaster* z opcją *–E*. Polecenie to działa szybko, pod warunkiem, że używasz NNTP.

Druga metoda działa dokładnie tak jak domyślne wygasanie obsługiwane przez *mthreads*. Usuwa tylko te wpisy, które odnoszą się do artykułów o numerach niższych niż dolny znacznik w pliku *active*. Można ją włączyć opcją *–e*.

Trzecia strategia usuwa całą bazę i zbiera ponownie wszystkie artykuły. Można ją włączyć, używając opcji *–E3*.

Lista grup do wygaśnięcia jest podawana przez opcję *–F* w ten sam sposób jak powyżej. Jednak jeżeli *nnmaster* działa jako demon, musisz go unicestwić (używając opcji *–k*), zanim nastąpi czas wygaśnięcia i zaraz potem uruchomić oryginalne opcje. Dlatego poprawne polecenie uruchamiane w celu usunięcia nieaktualnych artykułów ze wszystkich grup za pomocą pierwszej metody wygląda następująco:

```
# nnmaster -kF ""
# nnmaster -lrC
```

Istnieje wiele innych znaczników, które regulują zachowanie *nn*. Jeżeli martwisz się o usuwanie złych artykułów lub ich gromadzenie, przeczytaj stronę podręcznika *nnmaster*.

nnmaster opiera się na znajdującym się w katalogu */var/lib/nn* pliku *GROUPS*. Jeżeli go nie ma, gdy *nnmaster* jest uruchamiany po raz pierwszy, tworzy się go. Plik ten zawiera dla każdej grupy wiersz rozpoczynający się od jej nazwy, po której opcjonalnie występuje znacznik czasowy i znaczniki. Możesz je edytować, by włączyć jakieś cechy danej grupy, ale nie możesz zmieniać kolejności pojawiania się grup. (Ich kolejność musi się zgadzać z wpisami w pliku (binarnym) *MASTER*). Dopuszczalne znaczniki i ich działanie są również szczegółowo opisane na stronach podręcznika *nnmaster*.

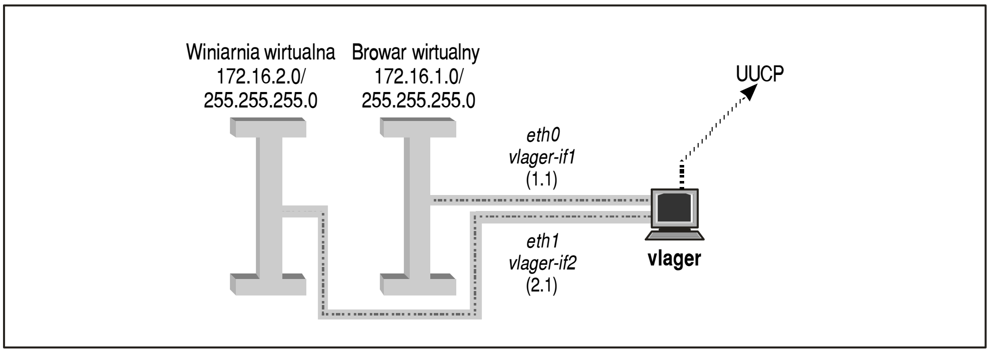
A
Przykładowa sieć: browar wirtualny

W tej książce posługiwaliśmy się poniższym przykładem, który jest nieco mniej skomplikowany od przykładu z uniwersytetem Groucho Marx i może być bardziej zbliżony do zadań, które rzeczywiście będziesz wykonywał.

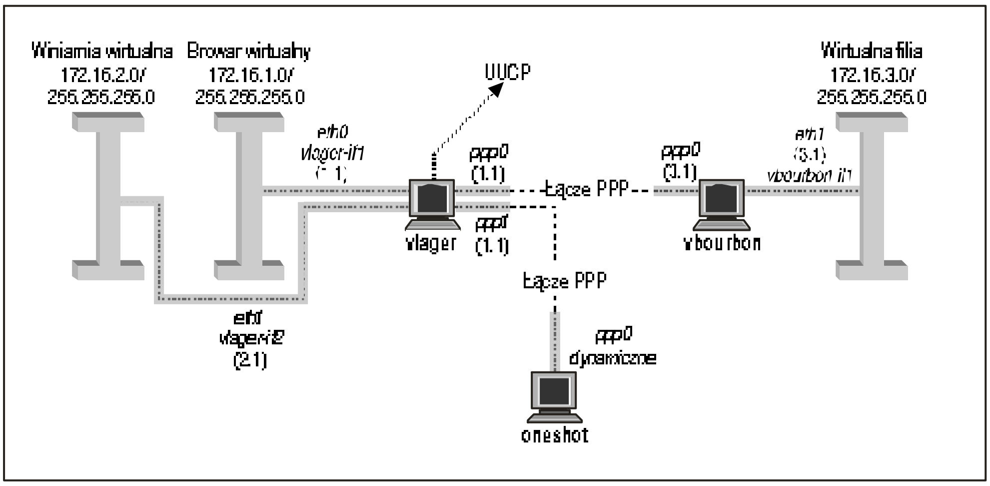
Browar wirtualny to niewielka firma, która jak sama nazwa wskazuje, zajmuje się warzeniem wirtualnego piwa. Aby efektywniej zarządzać swoim biznesem, właściciele browaru chcieli połączyć swoje komputery w sieć. Dobrze się złożyło, że są to PC, na których działa wspaniały Linux. Rysunek A-1 pokazuje konfigurację sieci.

Na tym samym piętrze znajduje się także winiarnia wirtualna, która ściśle współpracuje z browarem. Ma własną sieć Ethernet. Zupełnie naturalne jest, że obie firmy chcą połączyć swoje sieci. Pierwszym krokiem jest skonfigurowanie gatewaya, który przekazuje datagramy pomiędzy dwoma podsieciami. Dalej chcą mieć łącze UUCP ze światem zewnętrznym, przez które mogą wymieniać pocztę i grupy dyskusyjne. Na dłuższą metę będą także chciały zestawiać połączenia PPP w celu łączenia się z lokalizacjami odległymi i z Internetem.

Browar wirtualny i wirtualna winiarnia mają podsieci klasy C wydzielone z sieci B browaru, a gatewayem do każdej z nich jest host **vlager**, który obsługuje także połączenie UUCP. Konfigurację pokazano na rysunku A-2.



Rysunek A-1. Podsieci browaru i winiarni wirtualnej



Rysunek A-2. Sieć browaru wirtualnego

Podłączanie sieci wirtualnej filii

Wirtualny browar rozrasta się i otwiera filię w innym mieście. W filii działa oddzielna sieć Ethernet posiadająca własny numer IP sieci **172.16.3.0**, który jest 3. podsiecią sieci klasy B browaru. Host **vlager** działa jako gateway dla sieci browaru i obsługuje łącze PPP. Jego partner w nowej gałęzi to **vbourbon** posiadający adres IP **172.16.3.1**. Tę sieć pokazuje rysunek A-2.

B

Przydatne konfiguracje kabli

Jeśli chcesz połączyć ze sobą dwa komputery, a nie masz sieci Ethernet, potrzebujesz kabla szeregowego null modem lub kabla równoległego PLIP.

Kable te można kupić w sklepie, ale będzie dużo taniej i prościej, jeśli zrobisz je sam.

Kabel równoległy PLIP

Aby zrobić kabel równoległy używany do połączenia PLIP, będziesz potrzebował dwóch złączy 25-pinowych (zwanych DB-25) i kabla o przynajmniej jedenastu żyłach. Kabel nie może być dłuższy niż 15 metrów (50 stóp). Kabel może, ale nie musi być ekranowany, choć gdy robisz długi kabel, lepiej jest zastosować ekran.

Jeśli patrzysz na złącze, powinieneś zauważyć niewielkie numerki przy każdym pinie – od 1 dla pinu po lewej stronie u góry (jeżeli trzymasz szerszą stroną do góry) do 25 przy pinie po prawej stronie na dole. W przypadku kabla null printer musisz połączyć ze sobą odpowiednie piny obu złączy tak jak pokazano na rysunku B-1.

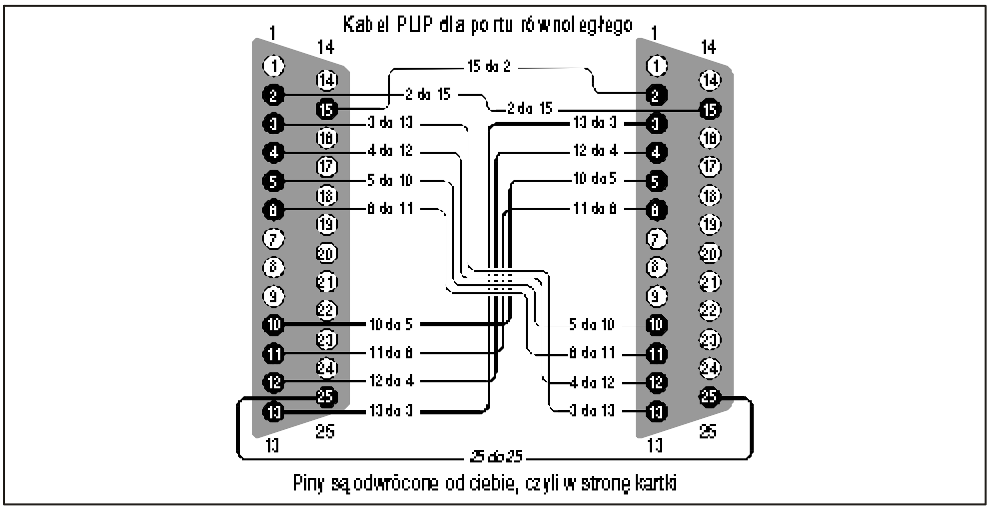
Wszystkie pozostałe piny należy pozostawić niepodłączone. Jeżeli kabel jest ekranowany, ekran powinien być podłączony do metalowej obudowy DB-25 tylko po *jednej* stronie.

Kabel szeregowy NULL modem

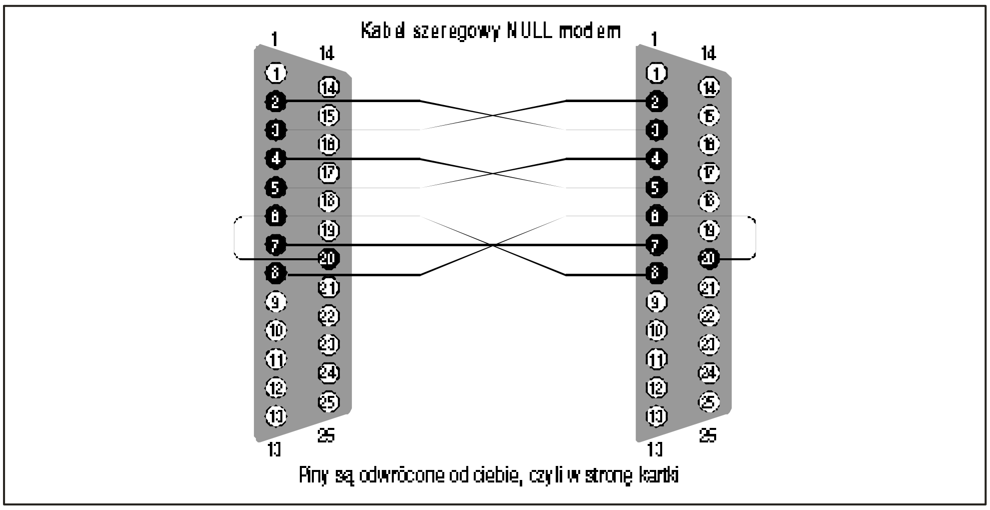
Kabel szeregowy null modem będzie działał zarówno dla połączenia SLIP, jak i dla PPP. Znów potrzebujesz dwóch złączy DB-25. Tym razem kabel musi być ośmiożyłowy.

Być może spotkałeś się z inną budową kabli null modem, ale ta pozwala ci na zastosowanie sprzętowej kontroli przepływu – co jest dużo lepsze od kontroli XON/XOFF – lub żadnej. Połączenia pokazano na rysunku B-2.

Znów, jeżeli masz kabel ekranowany, powinieneś podłączyć pierwszy pin tylko po jednej stronie.



Rysunek B-1. Kabel równoległy PLIP



Rysunek B-2. Kabel szeregowy NULL modem

C

Linux – podręcznik administratora sieci. Wydanie drugie*. Informacje o prawach autorskich

Wersja elektroniczna tej książki, która w czasie drukowania tej pozycji zawiera dokładnie tę samą treść co wersja drukowana O'Reilly'ego, jest dostępna na warunkach licencji GNU FDL. Prawa do przedruku dokumentu na Licencji FDL obejmują prawo do drukowania i rozpowszechniania drukowanych kopii wersji elektronicznej. Prawa do kopiowania drukowanej wersji O'Reilly są zastrzeżone. Elektroniczną wersję licencji można znaleźć pod adresem *http://www.oreilly.com/catalog/linag/licenseinfo.html*. Książka jest dostępna pod adresem *http://www.linuxdoc.org/LDP/nag/nag.html* oraz *http://www.oreilly.com/catalog/linag/* i może być zamieszczana w innych miejscach.

Zezwala się na kopiowanie, drukowanie, rozpowszechnianie i modyfikowanie dokumentu elektronicznego na warunkach licencji GNU Free Documentation License w wersji 1.1 lub jakiejkolwiek nowszej wersji opublikowanej przez Free Software Foundation. W przypadku Sekcji niezmiennych, takich jak podziękowania (we *Wstępie* i dodatku C pt. *Linux – podręcznik administratora sieci. Wydanie drugie. Informacje o prawach autorskich*), dalsze podziękowania można dodawać tylko poza tymi sekcjami. Na początku musi znaleźć się następująca informacja:

> *Linux – podręcznik administratora sieci*
> Olaf Kirch i Terry Dawson
> Copyright © 1993 Olaf Kirch
> Copyright © 2000 Terry Dawson
> Copyright wersji drukowanej O' Reilly'ego © 2000 O'Reilly & Associates

---

* W języku polskim jest to pierwsze wydanie tej książki (–przyp. red.).

Poniżej zamieszczono kopię Licencji GNU Free Documentation License, którą można znaleźć także (w wersji oryginalnej) pod adresem *http://www.gnu.org/copyleft/fdl.html*.

0. Preambuła

Celem niniejszej licencji jest zagwarantowanie wolnego dostępu do podręcznika, treści książki i wszelkiej dokumentacji w formie pisanej oraz zapewnienie każdemu użytkownikowi swobody kopiowania i rozpowszechniania wyżej wymienionych, z dokonywaniem modyfikacji lub bez, zarówno w celach komercyjnych, jak i niekomercyjnych. Ponadto Licencja ta pozwala przyznać zasługi autorowi i wydawcy przy jednoczesnym ich zwolnieniu z odpowiedzialności za modyfikacje dokonywane przez innych.

Niniejsza Licencja zastrzega też, że wszelkie prace powstałe na podstawie tego dokumentu muszą nosić cechę wolnego dostępu w tym samym sensie co produkt oryginalny. Licencja stanowi uzupełnienie Powszechnej Licencji Publicznej GNU (GNU General Public License), która jest licencją dotyczącą wolnodostępnego oprogramowania.

Niniejsza Licencja została opracowana z zamiarem zastosowania jej do podręczników do wolnodostępnego oprogramowania, ponieważ wolnodostępne oprogramowanie wymaga wolnodostępnej dokumentacji: wolnodostępny program powinien być rozpowszechniany z podręcznikami, których dotyczą te same prawa, które wiążą się z oprogramowaniem. Licencja ta nie ogranicza się jednak do podręczników oprogramowania. Można ją stosować do różnych dokumentów tekstowych, bez względu na ich przedmiot oraz niezależnie od tego, czy zostały opublikowane w postaci książki drukowanej. Stosowanie tej Licencji zalecane jest głównie w przypadku prac, których celem jest instruktaż lub pomoc podręczna.

1. Zastosowanie i definicje

Niniejsza Licencja stosuje się do podręczników i innych prac, na których umieszczona jest pochodząca od właściciela praw autorskich informacja, że dana praca może być rozpowszechniana wyłącznie na warunkach niniejszej Licencji. Używane poniżej słowo „Dokument" odnosić się będzie do wszelkich tego typu publikacji. Ich odbiorcy nazywani będą licencjobiorcami.

„Zmodyfikowana wersja" Dokumentu oznacza wszelkie prace zawierające Dokument lub jego część w postaci dosłownej bądź zmodyfikowanej i/lub przełożonej na inny język.

„Sekcją drugorzędną” nazywa się dodatek opatrzony odrębnym tytułem lub sekcję początkową Dokumentu, która dotyczy wyłącznie związku wydawców lub autorów Dokumentu z ogólną tematyką Dokumentu (lub zagadnieniami z nią związanymi) i nie zawiera żadnych treści bezpośrednio związanych z ogólną tematyką (na przykład, jeżeli Dokument stanowi w części podręcznik matematyki, Sekcja drugorzędna nie może wyjaśniać zagadnień matematycznych). Wyżej wyjaśniany związek może się natomiast wyrażać w aspektach historycznym, prawnym, komercyjnym, filozoficznym, etycznym lub politycznym.

„Sekcje niezmienne” to takie Sekcje drugorzędne, których tytuły są ustalone jako tytuły Sekcji niezmiennych w nocie informującej, że Dokument został opublikowany na warunkach Licencji.

„Treść okładki” to pewne krótkie fragmenty tekstu, które w nocie informującej, że Dokument został opublikowany na warunkach Licencji, są opisywane jako „do umieszczenia na przedniej okładce” lub „do umieszczenia na tylnej okładce”.

„Jawna” kopia Dokumentu oznacza kopię czytelną dla komputera, zapisaną w formacie, którego specyfikacja jest publicznie dostępna. Zawartość tej kopii może być oglądana i edytowana bezpośrednio za pomocą typowego edytora tekstu lub (w przypadku obrazów złożonych z pikseli) za pomocą typowego programu graficznego lub (w przypadku rysunków) za pomocą ogólnie dostępnego edytora rysunków. Ponadto kopia ta stanowi odpowiednie dane wejściowe dla programów formatujących tekst lub dla programów konwertujących do różnych formatów odpowiednich dla programów formatujących tekst. Kopia spełniająca powyższe warunki, w której jednak zostały wstawione znaczniki mające na celu utrudnienie dalszych modyfikacji przez czytelników, nie jest Jawna. Kopię, która nie jest „Jawna”, nazywa się „Niejawną”.

Przykładowe formaty kopii Jawnych to: czysty tekst ASCII bez znaczników, format wejściowy Texinfo, format wejściowy LaTeX, SGML lub XML wykorzystujące publicznie dostępne DTD, standardowy prosty HTML przeznaczony do ręcznej modyfikacji. Formaty niejawne to na przykład PostScript, PDF, formaty własne, które mogą być odczytywane i edytowane jedynie przez własne edytory tekstu, SGML lub XML, dla których DTD i/lub narzędzia przetwarzające nie są ogólnie dostępne, oraz HTML wygenerowany maszynowo przez niektóre procesory tekstu jedynie w celu uzyskania danych wynikowych.

„Strona tytułowa” oznacza, w przypadku książki drukowanej, samą stronę tytułową oraz kolejne strony zawierające informacje, które zgodnie z tą Licencją muszą pojawić się na stronie tytułowej. W przypadku prac w formatach nie posiadających strony tytułowej „Strona tytułowa” oznacza tekst pojawiający się najbliżej tytułu pracy, poprzedzający początek tekstu głównego.

2. Kopiowanie dosłowne

Licencjobiorca może kopiować i rozprowadzać Dokument komercyjnie lub niekomercyjnie, w dowolnej postaci, pod warunkiem zamieszczenia na każdej kopii Dokumentu treści Licencji, informacji o prawie autorskim oraz noty mówiącej, że do

Dokumentu ma zastosowanie niniejsza Licencja, a także pod warunkiem nie umieszczania żadnych dodatkowych ograniczeń, które nie wynikają z Licencji. Licencjobiorca nie ma prawa używać żadnych technicznych metod pomiarowych utrudniających lub kontrolujących czytanie lub dalsze kopiowanie utworzonych i rozpowszechnianych przez siebie kopii. Może jednak pobierać opłaty za udostępnianie kopii. W przypadku dystrybucji dużej liczby kopii Licencjobiorca jest zobowiązany przestrzegać warunków wymienionych w punkcie 3.

Licencjobiorca może także wypożyczać kopie na warunkach opisanych powyżej, a także wystawiać je publicznie.

3. Kopiowanie ilościowe

Jeżeli Licencjobiorca publikuje drukowane kopie Dokumentu w liczbie większej niż 100, a licencja Dokumentu wymaga umieszczenia Treści okładki, należy dołączyć kopie okładek, które zawierają całą wyraźną i czytelną Treść okładki: treść przedniej okładki, na przedniej okładce, a treść tylnej okładki, na tylnej okładce. Obie okładki muszą też jasno i czytelnie informować o Licencjobiorcy jako wydawcy tych kopii. Okładka przednia musi przedstawiać pełny tytuł; wszystkie słowa muszą być równie dobrze widoczne i czytelne. Licencjobiorca może na okładkach umieszczać także inne informacje dodatkowe. Kopiowanie ze zmianami ograniczonymi do okładek, dopóki nie narusza tytułu Dokumentu i spełnia opisane warunki, może być traktowane pod innymi względami jako kopiowanie dosłowne.

Jeżeli napisy wymagane na którejś z okładek są zbyt obszerne, by mogły pozostać czytelne po ich umieszczeniu, Licencjobiorca powinien umieścić ich początek (taką ilość, jaka wydaje się rozsądna) na rzeczywistej okładce, a pozostałą część na sąsiednich stronach.

W przypadku publikowania lub rozpowszechniania Niejawnych kopii Dokumentu w liczbie większej niż 100, Licencjobiorca zobowiązany jest albo dołączyć do każdej z nich Jawną kopię czytelną dla komputera, albo wymienić w lub przy każdej kopii Niejawnej publicznie dostępną w sieci komputerowej lokalizację pełnej kopii Jawnej Dokumentu, bez żadnych informacji dodanych – lokalizację, do której każdy użytkownik sieci miałby bezpłatny anonimowy dostęp za pomocą standardowych publicznych protokołów sieciowych. W przypadku drugim Licencjobiorca musisz podjąć odpowiednie środki ostrożności, by wymieniona kopia Jawna pozostała dostępna we wskazanej lokalizacji przynajmniej przez rok od momentu rozpowszechnienia ostatniej kopii Niejawnej (bezpośredniego lub przez agentów albo sprzedawców) danego wydania.

Zaleca się, choć nie wymaga, aby przed rozpoczęciem rozpowszechniania dużej liczby kopii Dokumentu, Licencjobiorca skontaktował się z jego autorami celem uzyskania uaktualnionej wersji Dokumentu.

4. Modyfikacje

Licencjobiorca może kopiować i rozpowszechniać Zmodyfikowaną wersję Dokumentu na zasadach wymienionych powyżej w punkcie 2 i 3 pod warunkiem ścisłego przestrzegania niniejszej Licencji. Zmodyfikowana wersja pełni wtedy rolę Dokumentu, a więc Licencja dotycząca modyfikacji i rozpowszechniania Zmodyfikowanej wersji przenoszona jest na każdego, kto posiada jej kopię. Ponadto Licencjobiorca musi w stosunku do Zmodyfikowanej wersji spełnić następujące wymogi:

A. Użyć na Stronie tytułowej (i na okładkach, o ile istnieją) tytułu innego niż tytuł Dokumentu i innego niż tytuły poprzednich wersji (które, o ile istniały, powinny zostać wymienione w Dokumencie, w sekcji Historia). Tytułu jednej z ostatnich wersji Licencjobiorca może użyć, jeżeli jej wydawca wyrazi na to zgodę.

B. Wymienić na Stronie tytułowej, jako autorów, jedną lub kilka osób albo jednostek odpowiedzialnych za autorstwo modyfikacji Zmodyfikowanej wersji, a także przynajmniej pięciu spośród pierwotnych autorów Dokumentu (wszystkich, jeśli było ich mniej niż pięciu).

C. Umieścić na Stronie tytułowej nazwę wydawcy Zmodyfikowanej wersji.

D. Zachować wszelkie noty o prawach autorskich zawarte w Dokumencie.

E. Dodać odpowiednią notę o prawach autorskich dotyczących modyfikacji obok innych not o prawach autorskich.

F. Bezpośrednio po notach o prawach autorskich, zamieścić notę licencyjną zezwalającą na publiczne użytkowanie Zmodyfikowanej wersji na zasadach niniejszej Licencji w postaci podanej w Załączniku poniżej.

G. Zachować w nocie licencyjnej pełną listę Sekcji niezmiennych i wymaganych Treści okładki podanych w nocie licencyjnej Dokumentu.

H. Dołączyć niezmienioną kopię niniejszej Licencji.

I. Zachować sekcję zatytułowaną „Historia” oraz jej tytuł i dodać do niej informację dotyczącą przynajmniej tytułu, roku publikacji, nowych autorów i wydawcy Zmodyfikowanej wersji zgodnie z danymi zamieszczonymi na Stronie tytułowej. Jeżeli w Dokumencie nie istnieje sekcja pod tytułem „Historia”, należy ją utworzyć, podając tytuł, rok, autorów i wydawcę Dokumentu zgodnie z danymi zamieszczonymi na stronie tytułowej, a następnie dodając informację dotyczącą Zmodyfikowanej wersji, jak opisano w poprzednim zdaniu.

J. Zachować wymienioną w Dokumencie (jeśli taka istniała) informację o lokalizacji sieciowej, publicznie dostępnej Jawnej kopii Dokumentu, a także o podanych w Dokumencie lokalizacjach sieciowych poprzednich wersji, na których został on oparty. Informacje te mogą się znajdować w sekcji „Historia”. Zezwala się na pominięcie lokalizacji sieciowej prac, które zostały wydane przynajmniej cztery lata przed samym Dokumentem, a także tych, których pierwotny wydawca wyraża na to zgodę.

K. W każdej sekcji zatytułowanej „Podziękowania” lub „Dedykacje” zachować tytuł i treść, oddając również ton każdego z podziękowań i dedykacji.

L. Zachować wszelkie Sekcje niezmienne Dokumentu w niezmienionej postaci (dotyczy zarówno treści, jak i tytułu). Numery sekcji i równoważne im oznaczenia nie są traktowane jako należące do tytułów sekcji.

M. Usunąć wszelkie sekcje zatytułowane „Adnotacje". Nie muszą one być załączane w Zmodyfikowanej wersji.

N. Nie nadawać żadnej z istniejących sekcji tytułu „Adnotacje" ani tytułu pokrywającego się z jakąkolwiek Sekcją niezmienną.

Jeżeli Zmodyfikowana wersja zawiera nowe sekcje początkowe lub dodatki stanowiące Sekcje drugorzędne i nie zawierające materiału skopiowanego z Dokumentu, Licencjobiorca może je lub ich część oznaczyć jako sekcje niezmienne. W tym celu musi on dodać ich tytuły do listy Sekcji niezmiennych zawartej w nocie licencyjnej Zmodyfikowanej wersji. Tytuły te muszą być różne od tytułów pozostałych sekcji.

Licencjobiorca może dodać sekcję „Adnotacje", pod warunkiem, że nie zawiera ona żadnych treści innych niż adnotacje dotyczące Zmodyfikowanej wersji – mogą to być na przykład stwierdzenia o recenzji koleżeńskiej albo o akceptacji tekstu przez organizację jako autorytatywnej definicji standardu.

Na końcu listy Treści okładki w Zmodyfikowanej wersji, Licencjobiorca może dodać fragment „do umieszczenia na przedniej okładce" o długości nie przekraczającej pięciu słów, a także fragment o długości do 25 słów „do umieszczenia na tylnej okładce". Przez każdą jednostkę (lub na mocy ustaleń przez nią poczynionych) może zostać dodany tylko jeden fragment z przeznaczeniem na przednią okładkę i jeden z przeznaczeniem na tylną. Jeżeli Dokument zawiera już treść okładki dla danej okładki, dodaną uprzednio przez Licencjobiorcę lub w ramach ustaleń z jednostką, w imieniu której działa Licencjobiorca, nowa treść okładki nie może zostać dodana. Dopuszcza się jednak zastąpienie poprzedniej treści okładki nową pod warunkiem wyraźnej zgody poprzedniego wydawcy, od którego stara treść pochodzi.

Niniejsza Licencja nie oznacza, iż autor (autorzy) i wydawca (wydawcy) wyrażają zgodę na publiczne używanie ich nazwisk w celu zapewnienia autorytetu jakiejkolwiek Zmodyfikowanej wersji.

5. Łączenie dokumentów

Licencjobiorca może łączyć Dokument z innymi dokumentami wydanymi na warunkach niniejszej Licencji, na warunkach podanych dla wersji zmodyfikowanych w części 4 powyżej, jednak tylko wtedy, gdy w połączeniu zostaną zawarte wszystkie Sekcje niezmienne wszystkich oryginalnych dokumentów w postaci niezmodyfikowanej i gdy będą one wymienione jako Sekcje niezmienne połączenia w jego nocie licencyjnej.

Połączenie wymaga tylko jednej kopii niniejszej Licencji, a kilka identycznych Sekcji niezmiennych może zostać zastąpionych jedną. Jeżeli istnieje kilka Sekcji niezmiennych o tym samym tytule, ale różnej zawartości, Licencjobiorca jest zobowiązany uczynić tytuł każdej z nich unikalnym poprzez dodanie na jego końcu, w nawiasach, nazwy oryginalnego autora lub wydawcy danej sekcji, o ile jest znany, lub unikalne-

go numeru. Podobne poprawki wymagane są w tytułach sekcji na liście Sekcji niezmiennych w nocie licencyjnej połączenia.

W połączeniu Licencjobiorca musi zawrzeć wszystkie sekcje zatytułowane „Historia" z dokumentów oryginalnych, tworząc jedną sekcję „Historia". Podobnie ma postąpić z sekcjami „Podziękowania" i „Dedykacje". Wszystkie sekcje zatytułowane „Adnotacje" należy usunąć.

6. Zbiory dokumentów

Licencjobiorca może utworzyć zbiór składający się z Dokumentu i innych dokumentów wydanych zgodnie z niniejszą Licencją i zastąpić poszczególne kopie Licencji pochodzące z tych dokumentów jedną kopią dołączoną do zbioru, pod warunkiem zachowania zasad Licencji dotyczących kopii dosłownych we wszelkich innych aspektach każdego z dokumentów.

Z takiego zbioru Licencjobiorca może wyodrębić pojedynczy dokument i rozpowszechniać go niezależnie na zasadach niniejszej Licencji, pod warunkiem zamieszczenia w wyodrębnionym dokumencie kopii niniejszej Licencji oraz zachowania zasad Licencji we wszystkich aspektach dotyczących dosłownej kopii tego dokumentu.

7. Zestawienia z pracami niezależnymi

Kompilacja Dokumentu lub jego pochodnych z innymi oddzielnymi i niezależnymi dokumentami lub pracami nie jest uznawana za Zmodyfikowaną wersję Dokumentu, chyba że odnoszą się do niej jako do całości prawa autorskie. Taka kompilacja jest nazywana zestawieniem, a niniejsza Licencja nie dotyczy samodzielnych prac skompilowanych z Dokumentem, jeśli nie są to pochodne Dokumentu.

Jeżeli do kopii Dokumentu odnoszą się wymagania dotyczące Treści okładki wymienione w części 3 i jeżeli Dokument stanowi mniej niż jedną czwartą całości zestawienia, Treść okładki Dokumentu może być umieszczona na okładkach zamykających Dokument w obrębie zestawienia. W przeciwnym razie Treść okładki musi się pojawić na okładkach całego zestawienia.

8. Tłumaczenie

Tłumaczenie jest uznawane za rodzaj modyfikacji, a więc Licencjobiorca może rozpowszechniać tłumaczenia Dokumentu na zasadach wymienionych w punkcie 4. Zastąpienie Sekcji niezmiennych ich tłumaczeniem wymaga specjalnej zgody właściciela prawa autorskiego. Dopuszcza się jednak zamieszczanie tłumaczeń wybranych lub wszystkich Sekcji niezmiennych obok ich wersji oryginalnych. Podanie tłumaczenia niniejszej Licencji możliwe jest pod warunkiem zamieszczenia także jej oryginalnej wersji angielskiej. W przypadku niezgodności pomiędzy zamieszczonym tłumaczeniem a oryginalną wersją angielską niniejszej Licencji moc prawną ma oryginalna wersja angielska.

9. Wygaśnięcie

Poza przypadkami jednoznacznie dopuszczonymi na warunkach niniejszej Licencji nie zezwala się Licencjobiorcy na kopiowanie, modyfikowanie, czy rozpowszechnianie Dokumentu ani też na cedowanie praw licencyjnych. We wszystkich pozostałych wypadkach każda próba kopiowania, modyfikowania lub rozpowszechniania Dokumentu albo cedowania praw licencyjnych jest nieważna i powoduje automatyczne wygaśnięcie praw, które licencjobiorca nabył z tytułu Licencji. Niemniej jednak w odniesieniu do stron, które już otrzymały od Licencjobiorcy kopie albo prawa w ramach niniejszej Licencji, licencje nie zostaną anulowane, dopóki strony te w pełni się do nich stosują.

10. Przyszłe wersje Licencji

W miarę potrzeby Free Software Foundation może publikować nowe poprawione wersje GNU Free Documenation License. Wersje te muszą pozostawać w duchu podobnym do wersji obecnej, choć mogą się różnić w szczegółach dotyczących nowych problemów czy zagadnień. Patrz *http://www.gnu.org/copyleft/*.

Każdej wersji niniejszej Licencji nadaje się wyróżniający ją numer. Jeżeli w Dokumencie podaje się numer wersji Licencji, oznaczający, iż odnosi się do niego podana „lub jakakolwiek późniejsza" wersja licencji, Licencjobiorca ma do wyboru stosować się do postanowień i warunków albo tej wersji, albo którejkolwiek wersji późniejszej opublikowanej oficjalnie (nie jako propozycja) przez Free Software Foundation. Jeśli Dokument nie podaje numeru wersji niniejszej Licencji, Licencjobiorca może wybrać dowolną wersję kiedykolwiek opublikowaną (nie jako propozycja) przez Free Software Foundation.

D

SAGE: cech administratorów systemu

Jeżeli pisząc artykuły do grupy *comp.os.linux.** i czytając dokumentację, nie uzyskujesz wszystkich potrzebnych informacji, być może czas dołączyć do grupy SAGE – cechu administratorów systemu sponsorowanego przez USENIX. Sage zabiega o to, aby administrowanie systemem zyskało rangę zawodu. SAGE zrzesza administratorów systemów i administratorów sieci i pomaga im rozwijać umiejętności zawodowe, zapewnia forum dzielenia się problemami i ich rozwiązywania oraz pozwala dyskutować użytkownikom, zarządom i producentom na tematy związane z administrowaniem systemem.

Do aktualnych inicjatyw SAGE należą:

- Wspólne sponsorowanie wraz z USENIX-em dorocznej konferencji dla administratorów systemów (LISA).
- Publikowanie *Job Descriptions for System Administrators*, redagowanej przez Tinę Darmohray, pierwszej serii praktycznych broszurek i przewodników po zasobach obejmujących zagadnienia i techniki związane z administrowaniem systemem.
- Tworzenie ośrodków archiwum, **ftp.sage.usenix.org**, gdzie są gromadzone referaty z konferencji administratorów systemu i dokumentacja związana z administrowaniem systemem.
- Tworzenie grup roboczych w obszarach istotnych dla administratorów systemów, jak zadania, publikacje, polityki, elektroniczne rozpowszechnianie informacji, edukacja, producenci i standardy.

Aby dowiedzieć się więcej na temat stowarzyszenia USENIX i jego specjalnej sekcji technicznej SAGE, skontaktuj się z biurem stowarzyszenia pod numerem telefonu (510) 528-8649 w Stanach Zjednoczonych lub za pośrednictwem poczty elektronicznej: *office@usenix.org*. Aby otrzymać elektroniczne informacje, napisz na adres: *info@usenix.org*. Roczna składka dla członków SAGE to 25 USD (musisz być także członkiem USENIX). Członkowie są uprawnieni do bezpłatnego otrzymywania kwartalników technicznych „login:" i „Computing Systems" oraz mają rabaty przy rejestracji na konferencjach i sympozjach, a także przy zakupie publikacji i innych usług SAGE.

Indeks

D

E

F

G

H

I

O autorach

Olaf Kirch ma stopień naukowy w dziedzinie matematyki, ale zrezygnował z zajmowania się teorią kategorii i małych sieci ciągłych (category theory and compact continuous lattices) po uruchomieniu pierwszej wersji jądra Linuksa w 1992 roku. Żywo wspomina, z jaką radością uczył się Uniksa poprzez czytanie kodu źródłowego jądra Linuksa.

Od tego czasu Olaf uczestniczył w różnych projektach związanych z Linuksem, włączając w to pisanie dużych części jego implementacji NFS i uruchomienie, wraz z Jeffem Uphoffem, pierwszej pocztowej listy dyskusyjnej o bezpieczeństwie Linuksa w 1995 roku.

Aktualnie pracuje w firmie Caldera Systems, gdzie jest odpowiedzialny za rzeczy związane z sieciami oraz zagadnienia bezpieczeństwa, a czasami zastanawia się, czy to wszystko mu się śni, czy jest prawdą.

Wolne chwile lubi spędzać z Maren i córką Jule. A jeżeli czytałeś jego biografię w pierwszym wydaniu *Podręcznika administratora sieci*, to zmieniło się tyle, że Olaf obecnie ma prawo jazdy.

Terry Dawson jest operatorem radia amatorskiego i od dłuższego czasu entuzjastą Linuksa. Jest autorem wielu dokumentów HOWTO związanych z siecią, stworzonych w ramach projektu dokumentacji Linuksa. Aktywnie uczestniczy w szeregu innych projektów związanych z Linuksem.

Terry ma 15-letnie doświadczenie zawodowe w telekomunikacji. Obecnie zajmuje się badaniami nad zarządzaniem siecią w Telstra Research Laboratories. Mieszka w Sydney z żoną Maggie i synem Jackiem.